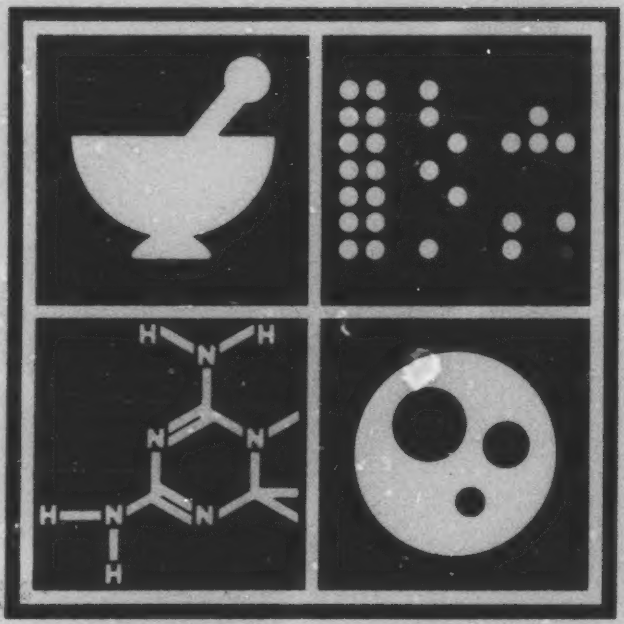
JUNE 86- 11392
Box 86-65



PROPHET

MOLECULES

U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES

Public Health Service
National Institutes of Health

P94/4/985

PROPHET MOLECULES

A User's Guide to the Molecule Facilities
of the PROPHET System

DIVISION OF RESEARCH RESOURCES
NATIONAL INSTITUTES OF HEALTH

Prepared by Bolt Beranek and Newman Inc., Cambridge, Mass.
Under Contract # N01-RR-5-2101
Biomedical Research Technology Program
Division of Research Resources
National Institutes of Health

REVISED SEPTEMBER 1985

NEWARK PUBLIC LIBRARY
MAR 14 1986
U. S. GOVT. PUBS.
DEPOSITORY 376

HE 20. 3008:

TABLE OF CONTENTS

Page

List of Tables

List of Figures

I. INTRODUCTION 1-1

II. MOLECULE COMMANDS 2-1

1. Making and Examining a Molecule 2-3

2. Computing and Changing a Molecule's Model . . . 2-21

3. Displaying a Molecule 2-41

4. Multimolecules 2-61

5. Disk Molecules and Multimolecules 2-75

6. Representation of Empirical Chemical Formulas . 2-81

III. MOLECULE FUNCTIONS AND PROCEDURES 3-1

1. The Molecule Data Type 3-3

2. Molecule Gatekeeper Procedures 3-15
   2A. ATOMS Table Gatekeepers 3-15
   2B. BONDS Table Gatekeepers 3-23
   2C. CONSTRAINTS Table Gatekeepers 3-27
   2D. ATTRIBUTES Gatekeepers 3-30
   2E. Miscellaneous Gatekeepers 3-32

3. Other Molecule Procedures 3-37
   3A. Making and Adjusting Molecules 3-38
   3B. Retrieving Chemical Information from System Tables 3-43
   3C. Computing Properties of a Molecule 3-46
   3D. Manipulating Molecular Models 3-54
   3E. Accessing and Modifying Constraints 3-66
   3F. Substructure Search Tools 3-72
   3G. Empirical Chemical Formulas 3-74
   3H. Setting Up Selector Columns 3-78

4. Multimolecules 3-83
   4A. The Structure of Multimolecules 3-83
   4B. Multimolecule Gatekeeper Procedures 3-85
   4C. The Multimolecule Quiz and INVOKE 3-93
   4D. Implementation of Multimolecules 3-96

5. Examples of Molecule Programming 3-101

READEX FICHE best copy available

Page

IV. VECTOR, MATRIX AND TRANSFORMATION PROCEDURES AND FUNCTIONS 4-1

1. Representation of Vectors and Matrices 4-3

2. Algebraic Operators 4-5

3. Vector and Matrix Functions 4-9

4. Transformers 4-13

5. Examples of VECTOR and MATRIX Uses 4-21

V. PUBLIC MOLECULE PROCEDURES 5-1

1. BESTPLANE 5-5
2. CLEAVEMOL 5-7
3. CONNECT 5-13
4. CONNLIST 5-19
5. CRYST 5-21
6. DISTAB 5-25
7. EDITMODEL 5-31
8. FITMOL 5-47
9. GENMOL 5-51
10. MAKEPOLYPEPTIDE 5-54.1
11. MAST 5-55
12. MBLD 5-57
13. MFBADDBONDS 5-61
14. MODIFY 5-63
15. MOLECYCLE 5-67
16. MOLEDAT 5-69
17. NEWMAN 5-73
18. ORMOL 5-77
19. PLANE 5-81
20. PRECONNECT 5-85
21. REFINEMOL 5-89
22. SETATOMNAMES 5-91
23. SETMOLNAMES 5-93
24. SETPEPTIDEBONDS 5-95
25. SPACEMOL 5-103
26. STEREOBOX 5-107
27. TABS2MOL 5-113

Appendices

A. Public Tables Containing Chemical Information A-1

B. An Introduction to Medical Foundation of Buffalo Molecule Tables and Procedures B-1

C. PROPHET Citations C-1

Index

LIST OF TABLES

Page

Table 3-1. Components of Molecule Structure 3-2

Table 3-2. Structure of the ATOMS table 3-5

Table 3-3. Structure of the BONDS table 3-7

Table 3-4. Structure of the CONSTRAINTS table 3-9

Table 3-5. System-recognized CONSTRAINT types 3-10

Table 3-6. ATOMS table gatekeeper procedures 3-16

Table 3-7. BONDS table gatekeeper procedures 3-24

Table 3-8. CONSTRAINTS table gatekeeper procedures 3-26

Table 3-9. ATTRIBUTES gatekeeper procedures 3-31

Table 3-10. Miscellaneous molecule gatekeeper procedures 3-33

Table 3-11. Procedures that make and adjust molecules 3-39

Table 3-12. Procedures that retrieve chemical information from system tables 3-44

Table 3-13. Procedures that compute properties of a molecule 3-47

Table 3-14. Procedures that manipulate molecular models 3-55

Table 3-15. Procedures used to set and retrieve constraints 3-67

Table 3-16. Procedures used in substructure search 3-73

Table 3-17. Procedures that handle empirical chemical formulas 3-75

Table 3-18. Procedures used to mark and unmark parts of molecules 3-79

Table 3-19. Multimolecule gatekeeper procedures 3-86

Table 3-20. Multimolecule quiz gatekeepers 3-94

Table 3-21. Multimolecule transformer gatekeepers 3-94

Table 4-1. VECTOR and MATRIX operations 4-6

Table 4-2. VECTOR and MATRIX functions 4-10

Table 4-3. Transformer operations and procedures 4-14

LIST OF FIGURES

Page

Figure 1-1. Display screen after sketching molecule in MAKE MOLECULE quiz 1-1

Figure 1-2. Display of acetone sketch 1-3

Figure 1-3. Display of acetone three-dimensional model complete with automatically generated hydrogens 1-3

Figure 1-4. Display of acetone stereo pair 1-3

Figure 1-5. Perspective display of acetone 1-4

Figure 1-6. A typical multimolecule 1-4

Figure 2-1. Display screen at beginning of MAKE MOLECULE session 2-4

Figure 2-2. Display screen after steps 1 to 4 of MAKE MOLECULE example 2-5

Figure 2-3. Display screen after steps 5 to 7 of MAKE MOLECULE example 2-6

Figure 2-4. Display screen after steps 8 to 11 of MAKE MOLECULE example 2-7

Figure 2-5. Display screen after completion of MAKE MOLECULE example 2-8

Figure 2-6. Substructure search examples 2-19

Figure 2-7. ADD RING quiz 2-24

Figure 2-8. Wipke quiz 2-26

Figure 2-9. Newman projection of the three adjacent bonds demonstrating the angle specifications for a dihedral angle constraint 2-30

Figure 2-10. Relationship of angles to bonds about a central atom with three bonds 2-34

Figure 2-11. Relationship of angles to bonds about a central atom with four bonds 2-34

Figure 2-12. Illustration of meaning of CC and CW chirality constraints 2-37

Figure 2-13. Constraint example 4 2-39

Figure 2-14. Display of ACETONE SKETCH 2-42

Page

Figure 2-15. Display of ACETONE THREED 2-42

Figure 2-16. Sample use of DEFAULT MOLECULES quiz 2-44

Figure 2-17. Example molecule MOL1 displayed THREED COMPLETE 2-46

Figure 2-18. Display screen after marking the ring bonds in MOL1 and selectively displaying them 2-46

Figure 2-19. Display screen after marking the side chain atoms in MOL1 and selectively displaying them 2-47

Figure 2-20. ACETONE displayed THREED COMPLETE NOTYPES ATOMINDEX 2-51

Figure 2-21. ACETONE display after orienting atoms 4 and 5 along the Y-axis 2-52

Figure 2-22. ACETONE molecule displayed with atoms 4 and 5 oriented along Y-axis and atom 8 in YZ-plane 2-52

Figure 2-23. %PERSPECTIVE display of ACETONE 2-58

Figure 2-24. MAKE/MANIPULATE MULTIMOLECULE quiz light buttons 2-62

Figure 2-25. MAKE MULTIMOLE: user has specified that ACETONE is to be restored from SMALLSAVEDMOLS and added to TWOACETONE 2-64

Figure 2-26. ADDING MOLECULE light buttons in multimolecule quiz 2-65

Figure 2-27. Display screen when using INSTALL light button 2-66

Figure 2-28. Multimolecule quiz example after installing ACETONE 2-67

Figure 3-1. ATOMS table of ACETONE molecule 3-5

Figure 3-2. BONDS table of ACETONE molecule 3-7

Figure 3-3. CONSTRAINTS table of ACETONE molecule 3-9

Figure 3-4. Clockwise dihedral angle formed by bonds B1 and B3 when looking down the axis of bond B2 3-50

Figure 3-5. MOLDATA table used in %CONNECTMOL example 3-58

Page

Figure 3-6. MOLQ constructed in %CONNECTMOL example displayed THREED COMPLETE before calling %CONNECTMOL 3-59

Figure 3-7. Interaction in %CONNECTMOL example 3-60

Figure 3-8. %CONNECTMOL summary of connections found in example 3-59

Figure 3-9. %CONNECTMOL quiz 3-61

Figure 3-10. MOLQ displayed THREED COMPLETE after %CONNECTMOL 3-61

Figure 3-11. ACETONE displayed THREED COMPLETE, labeling atoms by column 1 3-82

1. Introduction

1. Introduction

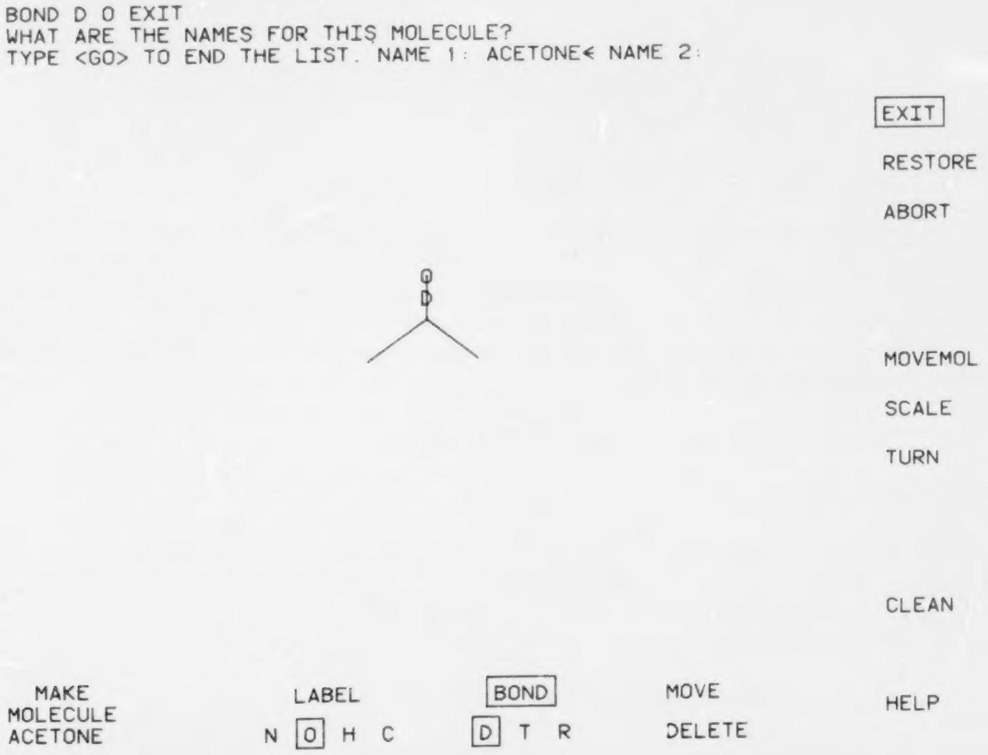
Chapter I.

INTRODUCTION

The PROPHET system contains a large assortment of facilities that enable you to enter, manipulate, and analyze chemical information. Chief among these is the MOLECULE data type that can contain complete information about the composition, structure and conformation of a molecule you are studying. You can create a molecule in PROPHET by simply typing a command such as:

```
MAKE MOLECULE ACETONE<GO>
```

on a display terminal and then sketching the molecule using the light pen and data tablet. The display screen at the end of one session of the MAKE MOLECULE quiz is shown in Figure 1-1. Chapter II, Part 1 explains in detail how to use the MAKE MOLECULE quiz to sketch a molecule.



```
BOND D O EXIT
WHAT ARE THE NAMES FOR THIS MOLECULE?
TYPE <GO> TO END THE LIST. NAME 1: ACETONE< NAME 2:
```

EXIT
RESTORE
ABORT
O
D
MOVEMOL
SCALE
TURN
CLEAN
MAKE
MOLECULE
ACETONE
LABEL
N O H C
BOND
D T R
MOVE
DELETE
HELP

Figure 1-1. Display screen after sketching molecule in MAKE MOLECULE quiz.

Molecule Commands

Chapter II describes not only the PROPHET commands for making molecules, but also those for manipulating and displaying molecules and molecular information. This chapter focuses on the specialized PROPHET tools relating to the use of the MOLECULE data type; it therefore makes no attempt to cover the basic principles of using PROPHET in general. You are not, however, required to have extensive knowledge of the PROPHET system before you can use the information presented here; the level of familiarity with PROPHET that you would gain by reading and carrying out the exercises in the PROPHET Primer should be adequate preparation for Chapter II.

Once you have sketched a molecule, you may have the system compute a simple three-dimensional model of the molecule's conformation by typing a command like

```
COMPUTE MODEL OF ACETONE<GO>
```

The use of the COMPUTE MODEL command is described in Chapter II, Part 2. This part also describes ways in which you can control and refine the model that the PROPHET system computes for a molecule.

Part 3 of Chapter II describes many of the ways in which a molecule can be displayed. For example, the command

```
DISPLAY ACETONE<GO>
```

produces the display of the sketch shown in Figure 1-2. The command

```
DISPLAY ACETONE THREED COMPLETE<GO>
```

produces the display of the three-dimensional conformation of a molecule complete with hydrogens shown in Figure 1-3. You can even get a display of a molecule as a stereo pair for three-dimensional viewing by typing a command like

```
DISPLAY ACETONE STEREO COMPLETE<GO>
```

as shown in Figure 1-4.

Part 3 of Chapter II also describes how you can freely reposition and orient the display of your molecule. In addition, you can get a special perspective display of your molecular model. The display produced by the command

```
CALL %PERSPECTIVEQK(ACETONE)
```

is shown in Figure 1-5.

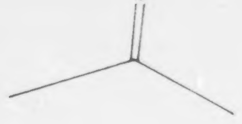


Figure 1-2. Display of acetone sketch.

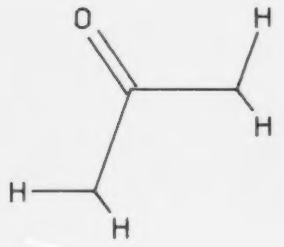


Figure 1-3. Display of acetone three-dimensional model complete with automatically generated hydrogens.

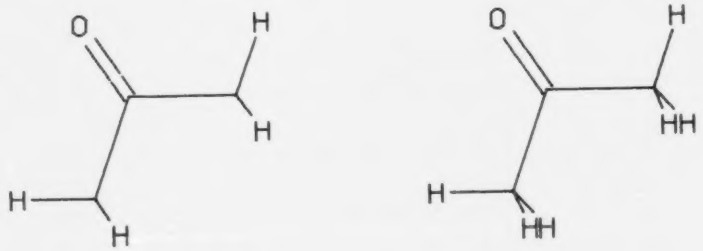


Figure 1-4. Display of acetone stereo pair.

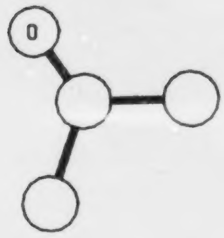


Figure 1-5. Perspective display of acetone.

Part 4 of Chapter II describes a data type known as a MULTIMOLECULE that you may use when you wish to study the spatial relationships of two or more molecules with three-dimensional models. One such multimolecule is illustrated in Figure 1-6.

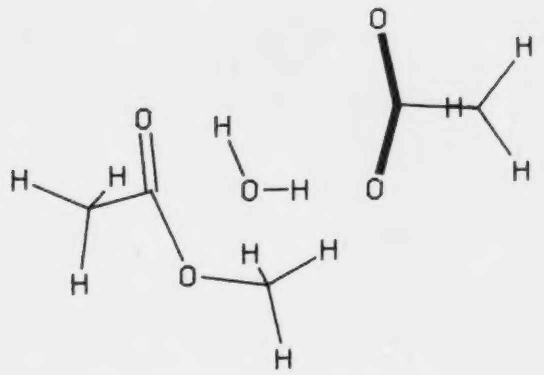


Figure 1-6. A typical multimolecule.

Part 5 of Chapter II provides a description of the PROPHET facility known as the disk molecule for handling very large molecules such as polypeptides. Finally, Part 6 is a brief description of the conventions PROPHET uses in calculating and ordering empirical chemical formulas.

Molecule Programming

In addition to the large body of PROPHET commands for handling chemical information, the PL/PROPHET programming language contains numerous tools for creating, manipulating, and analyzing molecules. Chapter III provides detailed information on the MOLECULE data type and describes a large assortment of PL/PROPHET functions and procedures that you may use in writing your own procedures. This chapter does not attempt to cover the basic features of the PL/PROPHET programming language, so you should be familiar with the *User's Guide to PL/PROPHET* before you attempt to use the information in Chapter III. In addition, you need to be familiar with at least the first three parts of Chapter II in order to take full advantage of the information in Chapter III.

Since the predominant use of vector and matrix operations in the PROPHET system is in the display of molecules and their three-dimensional models, Chapter IV describes the operators, functions and procedures that are available in PL/PROPHET for handling vectors, matrices, and transformations. Chapter IV assumes a thorough familiarity with the *User's Guide to PL/PROPHET* as well as some knowledge of the information in Chapter III.

Public Procedures

Chapter V contains descriptions of those PROPHET public procedures that are of use to the PROPHET molecule user. The public procedures are presented in alphabetical order so that you can find them easily, and a brief introduction lists them in five groups according to their functions. You need no more than a general familiarity with Chapter II in order to use the public procedures described in Chapter V.

Finally, the manual concludes with two short appendices. Appendix A includes reproductions of several public tables that contain information used by the PROPHET molecular model builder. Appendix B describes the public tables of molecular structures and the set of public molecule procedures that the Medical Foundation of Buffalo has contributed to the PROPHET system.

2. Molecule Commands

2. Molecule Commands

Chapter II.

MOLECULE COMMANDS

The purpose of this chapter is to acquaint you with the PROPHET facilities for creating and manipulating molecular structures. Part 1 describes one common method of creating a molecule in the PROPHET System that involves drawing a sketch of the molecule's topological structure. Part 2 presents details on how to use PROPHET to create an approximate three-dimensional model of the conformation of a molecule. In Part 3, methods for displaying molecules on the screen of your graphics terminal are described. Part 4 introduces the concept of a MULTIMOLECULE, a collection of molecules that may be manipulated in a common three-dimensional framework. We present the techniques available in PROPHET for handling very large molecules in Part 5. Finally, Part 6 describes some specialized PROPHET features for handling empirical chemical formulas according to the conventions followed by the Chemical Abstracts Service.

Part 1: Maki

The Sketch of a Molecule

A molecule in th
defined by drawing a
The sketch that you d
no explicit geome
conformation of the
information from your
are in the molecule a
one another. This i
molecule and is uniqu
takes a separate
coordinates of a mole
Only after this is
the atoms in space a

The MAKE MOLECULE Quiz

To define a mol

MAKE MOLECUL

A number of light bu
use to construct a
to select a particul
point to it or ty
User's Manual for a
tablet and light
screen.) In the foll
a simple example of
explain the use of

An Example

Suppose we wis
represents acetone
command:

MAKE MOLECU

After we press <GO>
display of light bu

PROPHET system is often first
rough chemical diagram or sketch.
aw to define the molecule conveys
rical information about the
molecule. The system obtains
sketch concerning only what atoms
d which atoms are connected to
s a topological description of the
e except for optical isomers. It
peration to compute the atomic
cule (see Part 2 of this chapter).
one is the relative arrangement of
sociated with the molecule.

cule, type the command

```
unused-name
```

tons then appear, which you may
sketch of the molecule. As usual,
r light button, you may either
e its name. (See Chapter 3 of the
escription of how to use the data
en to point to a position on the
wing paragraphs, we first present
he MAKE MOLECULE quiz, and then we
ach light button in some detail.

to make a PROPHET molecule that
We could begin by typing the

```
E ACETONE<GO>
```

the screen is erased and the
tons shown in Figure 2-1 is shown.

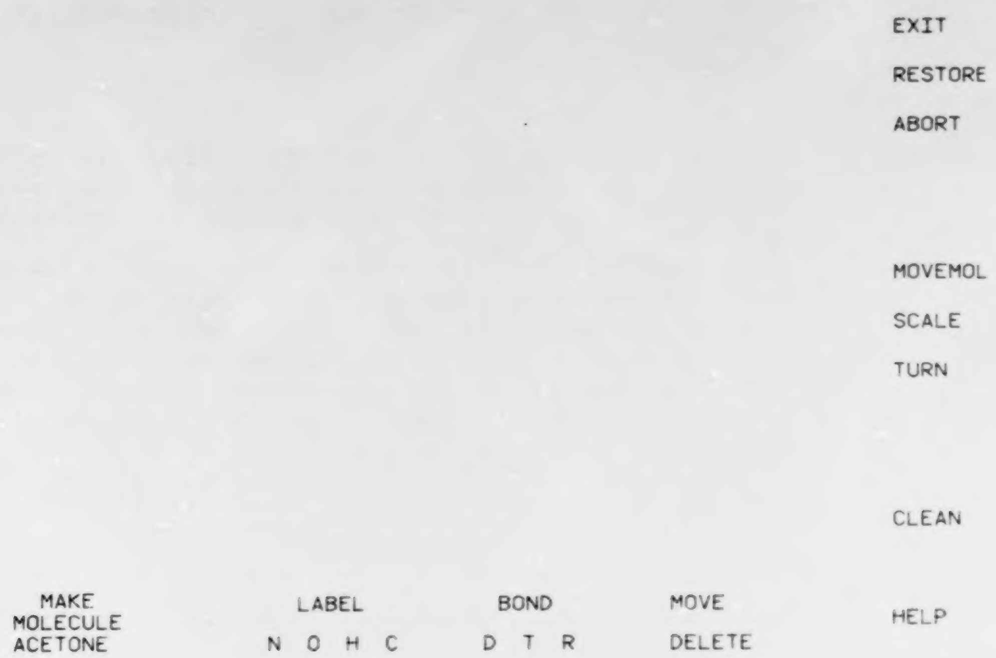


Figure 2-1. Display screen at beginning of MAKE MOLECULE session.

There are many ways in which we could use these light buttons to successfully construct our acetone molecule. We will describe one way here; you may wish to try following this example step-by-step at your terminal, after which you may experiment with using other sequences of light buttons to make larger or different test molecules. We could make the sketch by pointing to the following screen positions in this order:

Example: Defining Bonds

1. It is often convenient to begin by drawing bonds in order to quickly sketch out the basic layout of the molecule. To do this, we point first to the BOND light button. PROPHET draws a rectangle around the light button and then types the word BOND at the top of the screen to inform us that we are in bond-drawing mode.

2. Now we can actually begin to draw the molecule. We point to a position in the empty area of the screen a little low and to the left of the center of the screen. At the position we point to, a dot appears on the screen. This is one end of a bond and becomes the position of the first atom in our molecule.

3. Now we point to a position at about the center of the screen. A dot also appears there. Then a line appears between this dot and the dot we drew in step 2. This line represents a single bond joining two carbon atoms (the dots). We could later use other light buttons to specify other elements or bond types.

4. We may continue drawing bonds by continuing to point; we point this time to a position to the lower right of the center of the screen. Again a dot appears and a line is drawn between this dot and the last dot we pointed to. The display screen at this step is shown in Figure 2-2.

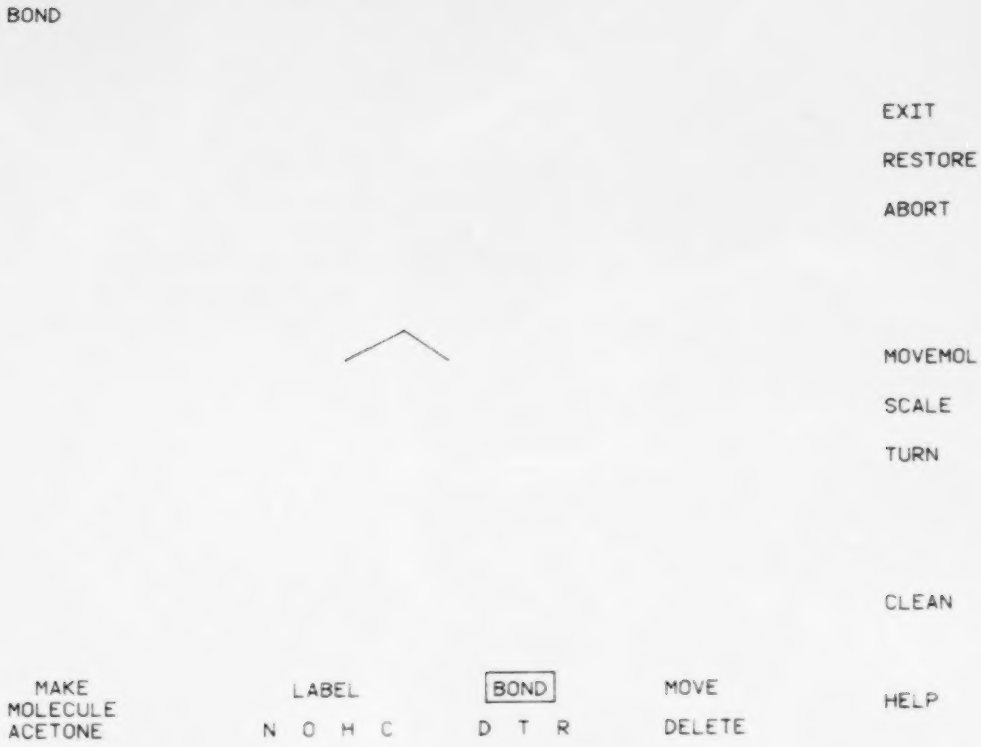

BOND
EXIT
RESTORE
ABORT
MOVEMOL
SCALE
TURN
CLEAN
MAKE
MOLECULE
ACETONE
LABEL
N O H C
BOND
D T R
MOVE
DELETE
HELP

Figure 2-2. Display screen after steps 1 to 4 of MAKE MOLECULE example.

Example:
Double
Bond

5. We have now defined the three carbon atoms in our acetone molecule, but we need to define the position of the double-bonded oxygen atom. We begin to do this by pointing to the D light button to indicate a double bond. PROPHET draws a box around the light button and types D at the top of the screen.

6. Now we point to the middle carbon atom to indicate where the double bond begins. While it is not necessary for us to hit the carbon atom precisely on the dot, we must be close enough so that it is clear that we are not sketching a new atom on the screen. We know that we have successfully pointed to the existing carbon atom when the dot representing that atom "flashes", i.e., momentarily gets brighter.

7. Now we point to the position of the oxygen atom directly above the carbon atom we just pointed to. A line is drawn between this position and the center carbon atom. PROPHET displays a D on this line to indcate that it represents a double bond. Figure 2-3 shows the screen at this point.

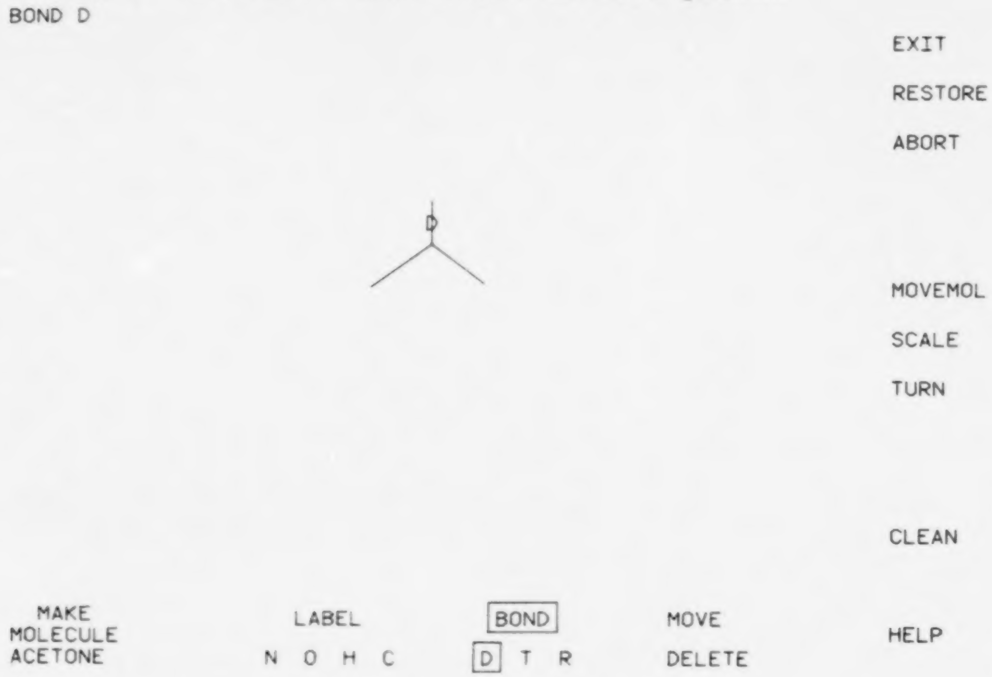


Figure 2-3. Display screen after Steps 5 to 7 of MAKE MOLECULE example.

Example: Labeling Atoms

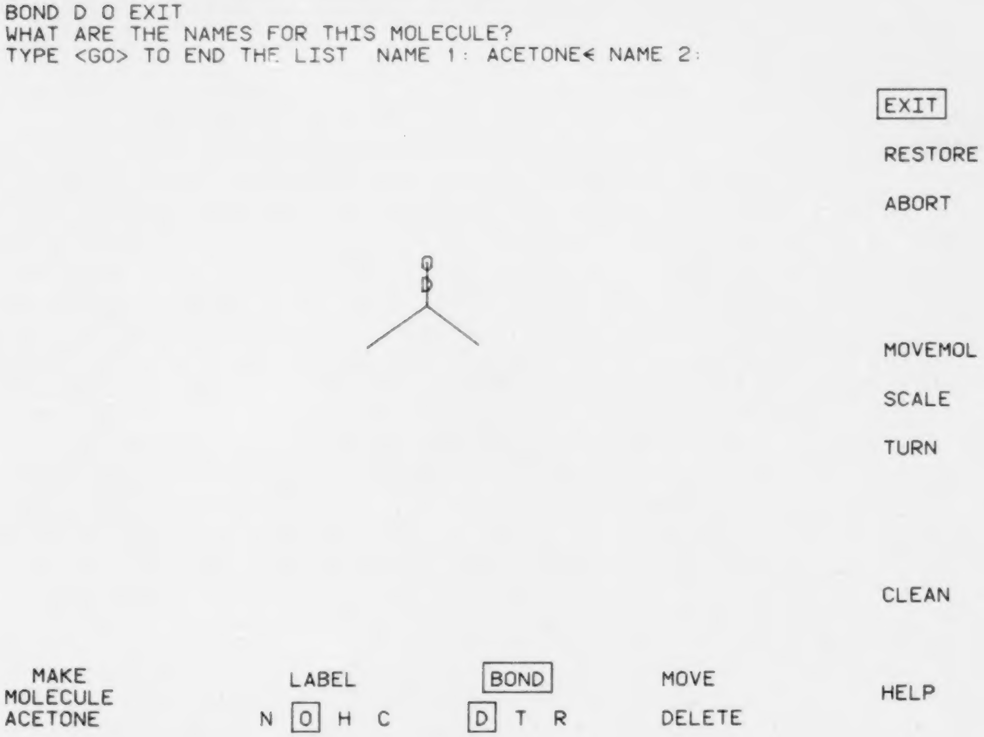
8. Since PROPHET automatically fills the valences of the carbon atoms with hydrogens, all we need to do to complete this sketch of the acetone molecule is to indicate that the atom at the end of the double bond is oxygen rather than carbon. We point to the O light button, a box is drawn around it and an O is typed at the top of the screen.

9. Now we point to the atom at the end of the double bond. The atom "flashes" and an O is displayed at that position.

Example: Finishing the Quiz

10. Since we are done sketching the molecule, we point to the EXIT light button. A box is drawn around EXIT and the word EXIT is typed at the top of the screen.

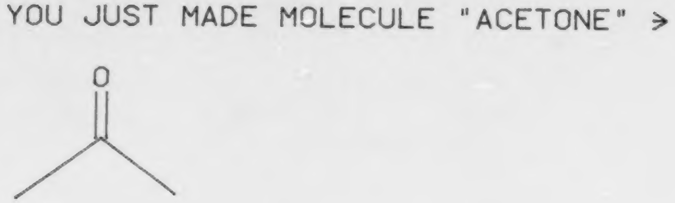
11. PROPHET asks what names we would like to have associated with the molecule. We respond by typing "ACETONE<GO>" when we are asked for Name 1 and just <GO> when we are asked for Name 2 as shown in Figure 2-4.



```
BOND D O EXIT
WHAT ARE THE NAMES FOR THIS MOLECULE?
TYPE <GO> TO END THE LIST  NAME 1: ACETONE< NAME 2:
```

Figure 2-4. Display screen after Steps 8 to 11 of MAKE MOLECULE example.

12. We are now finished making ACETONE. PROPHET erases the screen and redraws the molecule as shown in Figure 2-5.



YOU JUST MADE MOLECULE "ACETONE" >

Figure 2-5. Display screen after completion of MAKE MOLECULE example.

This example has shown how some of the light buttons are used in a typical session. The following paragraphs describe the use of each of the light buttons in the MAKE MOLECULE quiz in more detail.

BOND: Sketching Atoms and Bonds

Indicating BOND enables you to define a single bond between atoms by pointing to the endpoints of the bond. When you point to two previously defined atoms, a single bond is created between them (if the atoms are already bonded, the type of the bond is made single). If you indicate the endpoints where there are not atoms already, carbon atoms are automatically created there before the bond is completed. After pointing to the light button BOND, you may create several bonds in succession without pointing back to BOND. In that case, the second point of the last bond defined is taken to be the starting point of the next bond. When drawing a molecule in which more than two bonds are attached to a given atom, you will usually need to point to BOND more than once.

LABEL,N,O,H,C: Labeling Elements

All atoms are considered to be carbon atoms unless they are explicitly labeled otherwise. Use the N, O and H light buttons to label nitrogen, oxygen and hydrogen atoms. Simply point to the appropriate symbol, then point to one or more previously defined atoms. Each will be labeled with that symbol. To label an atom with any of the other acceptable symbols, point to the light button LABEL, type the standard chemical symbol needed, and then point to the atom or atoms you wish to have labeled.

PROPHET recognizes all of the standard chemical symbols found in the Periodic Table of the Elements. In addition, PROPHET provides the symbols "N+", "?" and "E". Use "N+" to indicate quaternary nitrogen, "?" when you wish to specify that any element could be in the position, and "E" to indicate free radicals as described below. Like the BOND light button, the LABEL light button remains activated until some other light button is indicated. To change from one symbol to another, however, it is necessary to indicate LABEL again, then type the new symbol. To relabel an atom to be a carbon, use the C light button.

Free Radicals

Hydrogen atoms attached to carbons and normal nitrogens need not be drawn; enough are assumed to satisfy the unused valence. If an atom of another type has a hydrogen on it, it must be indicated explicitly. In certain cases (e.g. a free radical), you may not wish hydrogens to be added automatically to fill the valence of a carbon or nitrogen atom. To handle these cases, attach a pseudo-atom to the atom in question and LABEL it with the symbol E. This will reserve one valence electron, but will add nothing further to the description of the molecule.

D,T,R: Labeling Bond Types

All bonds are considered to be single bonds unless specifically labeled otherwise. The D, T and R light buttons may be used to label double, triple and resonant bonds respectively. These buttons work in the same way as the ones for labeling an atom. It is usually easiest to use the BOND light button to draw the entire skeleton of the molecule, then label those atoms which are not carbons and those bonds which are not single. The D, T, or R light buttons can also be used to initially define the bond(s) in the same way as the BOND light button. This is particularly useful in defining a resonant ring structure. You should be aware that the PROPHET model builder can handle resonant bonds only in ring structures.

MOVE,DELETE,CLEAN: Changing the Sketch

The light buttons labeled MOVE and DELETE are used to change the sketch you have made. If an atom turns out to be in an awkward place, you can move it by pointing first to MOVE, then to the offending atom, then to the desired new position of the atom. This crosses out any old bonds connected to the atom, moves the atom, and redraws the bonds to its new position. You can move a bond as well by pointing to its midpoint. In this case, the bond is moved rigidly to the new position you point to by moving the atoms which are its endpoints. If you have made a mistake in drawing a bond or atom, you can delete it by pointing first to DELETE, then to the bond or atom. If an atom is deleted, all bonds leading to that atom are also deleted. All the deleted material is crossed out. This crossed out material can clutter up the display of your sketch. Pointing to the light button labeled CLEAN will regenerate the display without the crossed out material. Both the MOVE and DELETE light buttons remain in force after you have pointed to either of them until you point to another light button. Thus, more than one element can be moved or deleted in the same operation.

MOVEMOL: Moving the Sketch

The three light buttons along the right-hand side of the screen cause operations which affect the entire sketch at once. If you have begun drawing too near one edge of the screen, you may not have room to fit everything in. Pointing to the light button MOVEMOL allows you to move the entire sketch. This light button operates in a manner very similar to the MOVE command discussed in Chapter 3 of the User's Manual. You point first to the light button, then to some element in the sketch, then to the place on the screen where you wish that element to appear. The screen is erased and your sketch is redrawn in its new location. You may then continue to work on it.

SCALE: Scaling the Sketch

If you have started drawing too expansively at first, you may not have room to fit the entire molecule on the screen. Or, you may find you have drawn too small once the labels are attached to some of the atoms or bonds. By using the SCALE light button, you can either expand or contract the sketch. This light button operates in a manner very similar to the SCALE command discussed in Chapter 3 of the User's Manual. Point first to SCALE, then type a scale factor (a number). You may terminate this command by pressing <GO> or by pointing to the sketch (in the latter case, the sketch expands or shrinks around the indicated point). The default scale factor is 0.5. The screen is erased and a different-sized version of your sketch appears.

TURN: Turning the Sketch

You may decide that you prefer to have the sketch of the molecule in a different orientation from the one in which you entered it. To accomplish this, point to TURN, then type in an angle. You may terminate this command by pressing <GO> or by pointing to the sketch (in the latter case, the sketch is rotated around the indicated point). This light button works analogously to the TURN command described in Chapter 3 of the User's Manual, except that the only axis around which turns may be made is Z, perpendicular to the screen. The default angle is 90 degrees. The screen is erased and your sketch appears in the new orientation.

TURN and SCALE have a lasting effect on the molecule sketch. Whenever the sketch is displayed, it will appear with the orientation and size it had at the time you finished drawing it. On the other hand, the MOVEMOL light button affects the display only while you are drawing. The position of the molecule sketch when it is displayed later is determined by the considerations discussed in the User's Manual.

EXIT,RESTORE: After the Sketch is Completed

When you have finished drawing the sketch of the molecule, point to either EXIT or RESTORE. After a wait the system will ask:

```
WHAT ARE THE NAMES FOR THIS MOLECULE?
TYPE <GO> TO END THE LIST.  NAME 1:
```

The answers to this are simply text. Each name, e.g. the IUPAC name or common name, is terminated with the <GO> key. The system will continue to prompt you for names until you hit only the <GO> key in response. If no names are to be associated with the molecule, simply hit <GO>. Note that these names are part of the data associated with a variable of type molecule. They have no relation to the variable name you gave in the MAKE MOLECULE command, which is simply a handle for referring to the molecule within the system. If you later put the molecule into a table other than a SAVE table, the original variable name is stripped off.

The only difference between EXIT and RESTORE is that EXIT comes back with a new display of the molecule you have just defined, while RESTORE regenerates whatever PROPHET displays were on the screen when you first gave the MAKE MOLECULE command. When the MAKE MOLECULE command is finished, PROPHET types a helpful message like:

```
YOU JUST MADE MOLECULE "ACETONE"
```

Errors in the Sketch

Sometimes after you indicate EXIT or RESTORE, the system may return with an error message instead of requesting names. For example, PROPHET may type the message:

```
YOUR DISPLAY HAS ATOMS THAT AREN'T
CONNECTED TO YOUR MOLECULE
```

This error can arise when you delete an atom or bond, leaving the molecule in several unconnected parts (PROPHET insists that molecules be fully connected; however, you can use multimolecules, described below, to represent disconnected groups of atoms). In this case, PROPHET also flashes the unconnected parts of the molecule, so you can delete or connect them.

Another type of error results in the message:

```
THERE ARE n ATOMS OF UNKNOWN TYPE
IN YOUR MOLECULE
```

where n is the number of unknown atoms. This error occurs when you have used the LABEL light button to assign an unrecognized element label to an atom (e.g, CK instead of CL). The system flashes the offending atoms, so that you can relabel them, and then asks if you wish to make changes to correct the situation.

Another error that can be detected while making a molecule arises when an atom is connected to a set of bonds (like C=CL) which violate the legal bonding patterns recognized for it in the PROPHET System as listed in Appendix A of this manual. PROPHET tries to achieve nominal valences for carbons and nitrogens by <u>adding</u> hydrogens, but it can't help the =CL problem above. In such cases, the system types the message:

```
n ATOMS HAVE UNRECOGNIZED BOND PATTERNS
```

and flashes the offending atoms. It then asks:

```
DO YOU WISH TO MAKE CHANGES?
```

If you wish to keep the non-standard bonding patterns, type NO and the quiz will finish. If the error was the result of an actual mistake, typing YES allows you to reenter the quiz to correct it.

If you wish to quit at any point in the quiz, point to the ABORT light button. In that case, PROPHET asks you to verify your decision by typing YES or NO in response to the question:

```
DO YOU REALLY WANT TO ABORT?
```

Editing a Molecule

In some cases, you may find that you want to change a molecule that you had created using MAKE MOLECULE after you have exited from the quiz. You can do this by typing:

```
EDIT molecule-name
```

where molecule-name refers to the previously made molecule. The system reenters the MAKE MOLECULE quiz, but shows the molecule in its current state of development. All the MAKE MOLECULE commands can be used to modify the molecule's sketch as desired. Note that if you point to the ABORT light button (or type ABORT), the system asks you to verify your choice. If you use ABORT in this situation, the molecule is left in the state that it was in just before you typed EDIT molecule-name.

Saving Molecules

Once you have made a molecule, you may wish to preserve it so that you may use it in later sessions. You may do this by using the standard PROPHET SAVE command described in the User's Manual in the same way you may use SAVE to preserve graphs, procedures or any other temporary PROPHET variable. Molecules may be saved in your PRIVATE SAVED table, your project's SHARABLE SAVED table or in PRIVATE, PUBLIC, or SHARABLE named SAVE tables. When you wish to reuse a molecule, it is first necessary to use the RESTORE command as described in the User's Manual.

Making One Molecule from Another

If you are entering several structurally related molecules in PROPHET, you may wish to alter an existing molecule sketch to obtain another molecule rather than drawing in the same basic structure repeatedly. To accomplish this, use the command

```
MAKE MOLECULE unused-name FROM molecule-name
```

The screen is erased, the usual MAKE MOLECULE light buttons appear, and the sketch of the previously defined molecule is shown. You may add and delete bonds and atoms to create the new molecule desired without in any way affecting the already existing molecule. This command is terminated in the usual way, by indicating EXIT, RESTORE or ABORT.

Making One Molecule from Portions of Another

At times you may wish to make a new molecule that has much in common with some part of a molecule that you have already made. In such a case, you can indicate what parts of the old molecule are to be included in the new molecule and then start to make the new molecule by SELECTING these parts of the old molecule. The easiest way to indicate which items or bonds you are interested in is to use the MARK ATOMS or MARK BONDS commands described in Part 3 of this chapter for selective displays. Type:

```
     ATOMS
MARK          OF molecule-name FOR subset-name
     BONDS
```

where subset-name is a text string enclosed in single quotes. Then point to those atoms or bonds you wish to MARK. You may then use the command:

```
MAKE MOLECULE unused-name FROM molecule-name

              ATOMS
    SELECTING         BY subset-name
              BONDS
```

At this point, you are placed in the MAKE MOLECULE quiz, where you may make additions or changes to your new molecule. It is possible to select both atoms and bonds in making the same new molecule; if you do, the SELECTING ATOMS phrase must precede the SELECTING BONDS phrase in the command. You can also indicate which atoms and bonds are to be included in the new molecule directly in the MAKE MOLECULE ... FROM ... SELECTING ... command by using a fixed array of atom or bond indices in the SELECTING phrase. Details of how this command works are given in Part 3 of Chapter III in the description of the system procedure %MAKEMOLSELECT.

The NOQUIZ Option

When you are making a new molecule from an old one, or part of an old one, you may not wish to use the MAKE MOLECULE quiz to make any changes. In this case, use the NOQUIZ option:

```
MAKE MOLECULE unused-name FROM molecule-name NOQUIZ
```

or

```
MAKE MOLECULE unused-name FROM molecule-name

          ATOMS
SELECTING         BY subset-name NOQUIZ
          BONDS
```

If, for example, we were to make an unmodified copy of the ACETONE molecule we made in the MAKE MOLECULE example, we could type:

```
MAKE MOLECULE ACETONE2 FROM ACETONE NOQUIZ
```

We would then have ACETONE2 as an exact copy of ACETONE without going through the MAKE MOLECULE quiz.

Other Methods for Making Molecules

There are a number of PROPHET public procedures which you can use to make molecules without using the MAKE MOLECULE quiz. These are described in Chapter V. CRYST and CONNECT, for example, are public procedures that may be used in the process of constructing molecules from sets of coordinates, such as x-ray crystallographic coordinates. The public procedure MBLD allows construction of a molecule from a complete set of bond lengths, bond angles and torsion angles.

Accessing Other Parts of a Molecule

In addition to the sketch or model (see below) of a molecule, there are a few other pieces of information you might wish to look at. The following parts of a molecule may be accessed:

```
FORMULA
PERCOMP   OF molecule-name
NAMES
ATTRIBUTES
```

FORMULA of a Molecule

A molecule's FORMULA is of type CHEMTEXT, which is treated like text except that when it is displayed, the numbers are shown depressed half a line to give the appearance of a conventional chemical formula. CHEMTEXT formulas may be sorted according to the conventions used by the Chemical Abstracts Service as described later in this chapter. The FORMULA is normally computed by PROPHET from the sketch you have drawn. For example, the formula of ACETONE is C_3H_6O.

Per Cent Composition

The composition of a molecule can be referred to in PROPHET by the name PERCOMP. This is a TEXT describing the percentage (by weight) for each element type of the molecule, calculated from the FORMULA and a table of atomic weights. For example, if you enter the statement

```
TYPE PERCOMP OF ACETONE
```

where ACETONE is the name of a previously defined molecule, the following would appear:

```
CARBON            62.04%
HYDROGEN          10.41%
OXYGEN            27.55%

TOTAL WEIGHT      58.08
```

NAMES of a Molecule

NAMES is an array of TEXT strings containing different possible names for the compound. You enter these in response to a question at the end of the MAKE MOLECULE command. There will be no values in this array unless you enter them. (These can be added or changed later by using the SET command and accessing an individual element of the array. See the User's Manual for a discussion of arrays.) Note that you may change NAMES and FORMULA, but PERCOMP cannot be changed except by changing the composition of the molecule.

ATTRIBUTES of a Molecule

The ATTRIBUTES of a molecule are a set of user-supplied descriptors about the molecule (e.g., dipole moment, crystallographic cell constants, etc.) You can examine the ATTRIBUTES with the command:

```
TYPE ATTRIBUTES OF molecule-name
```

The ATTRIBUTES can be individually accessed and modified using system procedures described in Chapter III of this manual.

Searching for Molecular Structures

If you have a table which contains a column of molecules, you can search it for a particular molecule or for all the molecules containing a particular fragment or substructure. This is accomplished using the MAKE TABLE or DISPLAY command:

```
MAKE TABLE unused-name FROM table-name
FOR relational-expression
```

or

```
DISPLAY table-name FOR relational-expression
```

The "relational-expression" takes the form

```
column-reference CHEMCTNS molecule-name
```

The operator CHEMCTNS selects only rows of the table which contain the fragment named in the last argument.

To specify the fragment, use the command

```
MAKE MOLECULE unused-name
```

and draw the fragment in the usual fashion. Then use the LABEL light button to label attachment points with the symbol '?'. The specification of a '?' works something like the R notation for indicating a class of compounds. When the substructure search is carried out, the ? atom matches on any atom or group. Some examples are shown in Figure 2-6. The match between fragment and molecule includes all system-added hydrogens in both. Sometimes it is difficult to see why a fragment matches or doesn't match a particular molecule. This is usually due to the presence of hydrogens which have not been drawn in, but added by the system to carbon and nitrogen atoms. If no ? atoms are present in the fragment, only the exactly matching molecule is selected from the table.

Note that searching for molecular fragments is a relatively time-consuming procedure. In general, the more unspecified a fragment is, i.e. the more ? atoms there are, the longer it takes to determine whether or not the fragment matches a particular molecule. It may be more efficient, in some cases, to combine two relational expressions with the logical operator OR, than to use a loosely specified fragment. (See the User's Manual for a discussion of logical operators.) Before you undertake fragment searches of many molecular structures, you should speak with your PROPHET representative about the feasibility of your particular project.

Pre-Screening of Molecules

If you have a table with a column of molecules on which you plan to do a number of searches, it is possible to create a column, derived from the molecule column, which will help speed up the substructure search process. This is accomplished by adding a

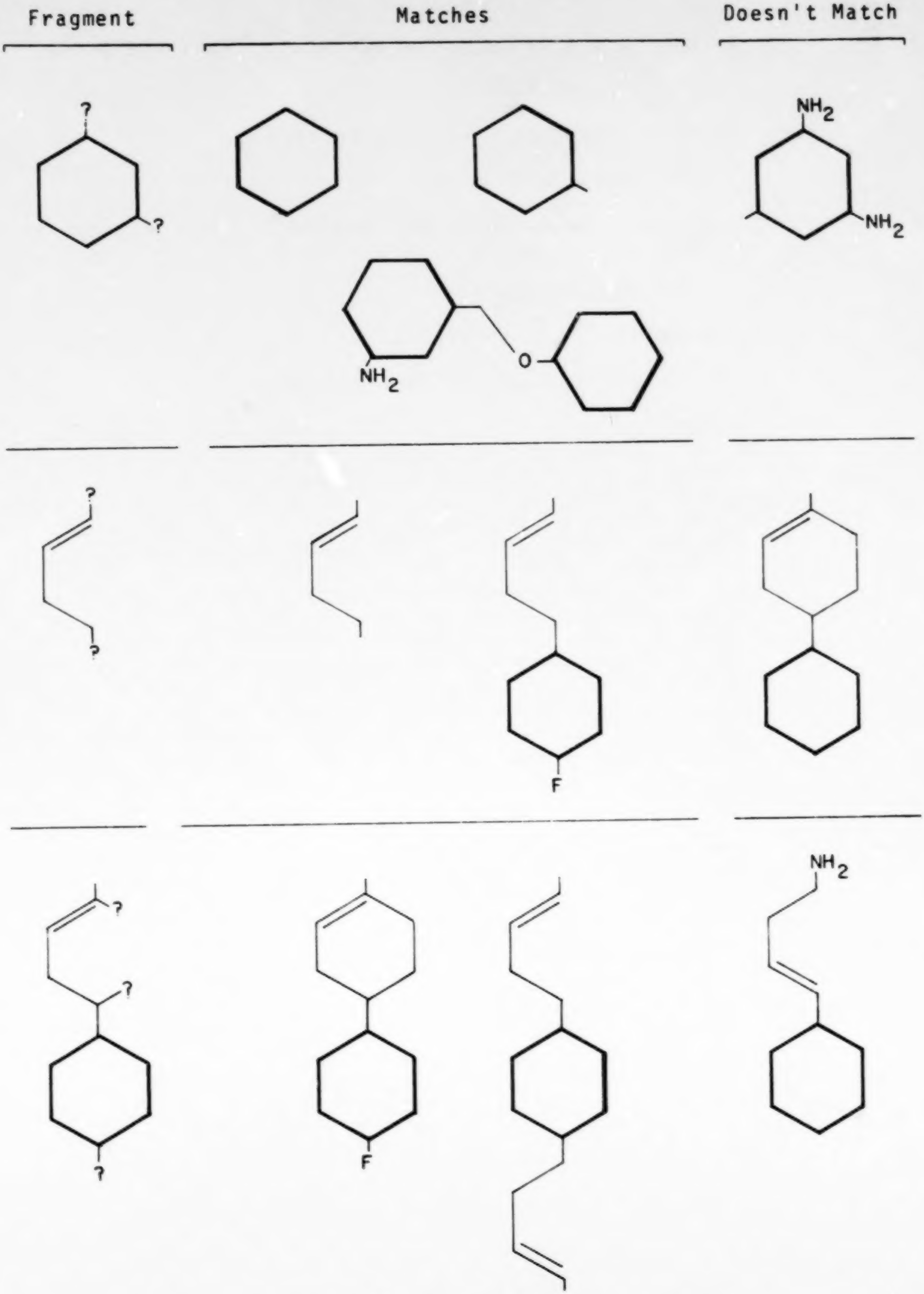

Fragment
Matches
Doesn't Match
?
?
NH2
NH2
NH2
O
?
?
F
?
?
?
F
NH2

Figure 2-6. Substructure search examples.

column to the table to be searched. When the prompt

```
COLUMN TYPE =
```

appears, you should respond by typing

```
MOLSCREEN
```

Then you type or point to DERIVED. The system will prompt

```
DERIVATION =
```

and you should respond with

```
SCREEN(CELL(column-reference))
```

where column-reference is the usual way of referring to a column in a derivation (see the User's Manual). SCREEN is a function which operates on an object of type MOLECULE and returns a value of type MOLSCREEN which contains information about the molecule which has been processed and reorganized in anticipation of your carrying out substructure searches on the molecule. The CELL notation is used in the ordinary way to refer to a column of the table in a derivation. The column reference may be, as usual, either the column-number or the column-heading (enclosed in single quotation marks). Once you have created a derived column of this type, all substructure searching can be carried out by using the derived column of type MOLSCREEN rather than the original column of molecules.

Molecular Conformation

An important step in creating a complete description of a molecule is defining its conformation. In PROPHET, the conformation is represented by a table containing the coordinates of the position of each atom in the molecule. While no completely satisfactory method has been found for automatically computing the conformation of an arbitrary molecule, the PROPHET system can build what is essentially a Dreiding model for a molecule you have defined. You can then use your chemical knowledge to improve the conformation.

The command to produce a conformational model of a molecule is

```
COMPUTE MODEL [OF] molecule-name
```

If the sketch of the molecule is currently displayed on the screen, you can point to it instead of typing in the molecule name. A conformation is then computed for the molecule using tables of standard bond lengths and bond angles. The public tables ELEMENTS, BONDLENGTHS and DIRECTIONS, which contain the standard bond length and bond angle information currently used by the PROPHET model builder, are reproduced in Appendix A of this manual.

Models Containing Ring Systems

Models of molecules containing rings or ring systems may be computed in two different ways. In the first way, PROPHET looks up the conformation of the ring in a ring dictionary. This, in principle, allows you to compute the model of any molecule if you first put some three-dimensional ring coordinates in one of the ring dictionaries (see below). In the second way of computing ring system models, PROPHET uses an energy minimization procedure developed by Dr. W. T. Wipke. This does not involve any prior knowledge of the conformation of a ring, but may sometimes fail to produce a reasonable conformation. Which method PROPHET uses is under your control. The ring dictionary method is automatically used first unless you specify otherwise. The energy minimization procedure is discussed further later in this part. In all the discussion that follows, the word "ring" refers to a ring or to a ring system. By "ring system" we mean a collection of rings for which each ring in the system has at least one atom in common with another ring in the system. Thus, a ring could be (1) a simple ring, (2) a fused ring, (3) a spiro ring, or (4) a bridged ring.

The Ring Dictionary

The first method PROPHET uses to compute models of molecules containing rings is to retrieve conformations for the rings from a *ring dictionary*. When PROPHET finds a ring in a molecule for which you requested a model, PROPHET carries out a substructure search to locate that ring in a ring dictionary. If the ring is not found, PROPHET uses the energy minimization procedure described below. The model builder has access to as many as three ring dictionaries, a PRIVATE, a SHARABLE, and a PUBLIC one. When the search is undertaken, the private one (if there is one) is searched first, the public one last. Thus, if you wish to have a ring with a different conformation from that in the public dictionary, you can enter it in your own private dictionary. The PUBLIC RINGS dictionary presently used in PROPHET is reproduced in Appendix A of this manual.

Looking at the Ring Dictionaries

Before you try to compute a model of a molecule containing a ring, you may want to see if a conformation already exists. To do this, use the command

```
        PUBLIC
DISPLAY SHARABLE RINGS display-qualifications
        PRIVATE
```

where the default PRIVATE is assumed, and all display qualifications applicable to tables may be used. As usual, PRINT or SCAN may be used in place of DISPLAY. In order to examine the conformation of a particular ring stored in a ring dictionary, you can use the system procedure %GETRINGMOL described in Part 3 of Chapter III.

The ADD RING Quiz

The ADD RING quiz may be used to add rings to your PRIVATE or SHARABLE ring dictionaries. If you wish to have a ring added to the PUBLIC ring dictionary, please speak with your PROPHET representative. To turn any MOLECULE into a valid ring dictionary entry, four things need to be done:

1. Trimming: The molecule must be "trimmed" so that there is exactly one ring system and so that all atoms and bonds are either ring atoms/bonds or exit atoms/bonds adjacent to a ring atom. In addition, the types of all exit atoms must be changed to '?'.

2. Conforming: All atoms in the trimmed molecule must be assigned 3D coordinates.

3. Assigning Priorities: All exit atoms may be assigned exit priorities as explained below.

4. The ring must be inserted in the appropriate ring dictionary together with its MOLSCREEN and a key that allows for efficient look-up.

To perform these tasks, use the following command:

```
         PRIVATE
     ADD          RING molecule-name
         SHARABLE
```

The ADD RING quiz creates a separate copy of the "source molecule" so that any changes made during the quiz do not affect the original molecule. This quiz performs the following steps:

1. The molecule specified by molecule-name is displayed. It is used as the basis of the ring entry.

2. The molecule is then checked for the presence of any ring system(s). If none are found, the program gives an error message and exits. If multiple ring systems are found, you are asked to select one by pointing at any atom or bond that is in the desired ring system.

3. Once a single ring system has been identified, the molecule is automatically trimmed to include just the ring system; that is, any atoms which are neither ring atoms/bonds or exit atoms/bonds are deleted. At this time, exit atoms are also converted to be of type '?'. The bonding patterns are checked for validity and the valences of ring atoms are balanced.

4. If 3D MODEL coordinates are not found in the original molecule for every atom in the trimmed molecule, ADD RING flashes the unlocated atoms and requests input of their [X Y Z] coordinates.

5. The quiz may then ask you to set "priorities" for the exit atoms. Priorities are integers assigned to exit atoms indicating which exit position (adjacent to a ring atom) receives which connected group when creating the model of a molecule. Priority 1 is the highest. Higher priority (lower number) exits get larger attached groups. Priority is only relevant around a ring atom when there are two or more exit bonds of the same type. If PRIORITIES are not yet specified for every exit atom, and the user wishes to enter PRIORITIES, the program cycles through the unassigned exit atoms, requesting input of their PRIORITY numbers. (If you do not care about the priority of a particular atom, you can leave it empty).

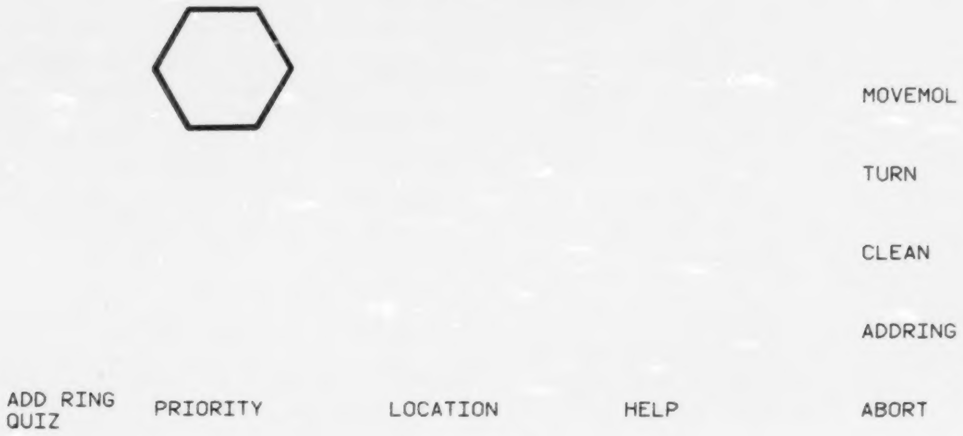


Figure 2-7. ADD RING quiz .

6. At this point, the program redisplays the trimmed molecule in THREED COMPLETE mode, using the augmented label feature described in Part 3 of this chapter to show the priorities of the exit atoms. A number of light buttons are then put up as shown in Figure 2-7 to allow further interactive modification. These light buttons may be used in the following ways:

a. PRIORITY: Asks for a new value for priority of the atom you point to.

b. LOCATION: Types out the current 3D location of the atom you point to and asks for the new coordinates.

c. MOVEMOL, SCALE, TURN and CLEAN: Operate like corresponding commands in the MAKE MOLECULE quiz.

d. ADDRING: Checks the current trimmed molecule for validity, computes a RING KEY and MOLSCREEN to store in the ring dictionary for use during model building and asks for a literature reference. ADDRING then puts the new entry into the specified ring dictionary. (If the dictionary did not exist before, it is created.) If the validity check fails, the user gets an error message and is returned to the interactive modification environment.

e. ABORT: Exits from the ADD RING quiz without adding to the ring dictionary.

The Energy Miminization Procedure

For molecules which contain rings, You may also have a model of the ring(s) computed by a classical energy minimization procedure developed by Dr. W. T. Wipke. If COMPUTE MODEL does not find a ring in one of the ring dictionaries, it invokes this energy minimization procedure. In order to perform this calculation, PROPHET asks you for some additional information (see below) about the ring(s) that it could not find in any of the ring dictionaries. If, for any reason, you do not wish to invoke the energy minimization procedure, you must give the command

```
SET %WIPKEFLAG TO FALSE
```

before you issue the COMPUTE MODEL command. If, on the other hand, you know that none of your rings are in any of the ring dictionaries, you can save some computation time by avoiding the search of the dictionaries by typing the command

```
SET %RINGSEARCH TO FALSE
```

before you type the COMPUTE MODEL command. This causes all the rings in the molecule to be computed using the energy minimization procedure (assuming that %WIPKEFLAG = TRUE). Remember to set %RINGSEARCH back to TRUE when you are finished if you plan to compute other molecular conformations.

The Wipke Quiz

If it is necessary to use the energy minimization procedure, you need to specify some additional information about the ring. The screen is erased and the molecule sketch is displayed. PROPHET flashes the bonds of each ring or ring system that the energy minimization procedure is being applied to. (Figure 2-8 shows the display after entry to the Wipke quiz.)

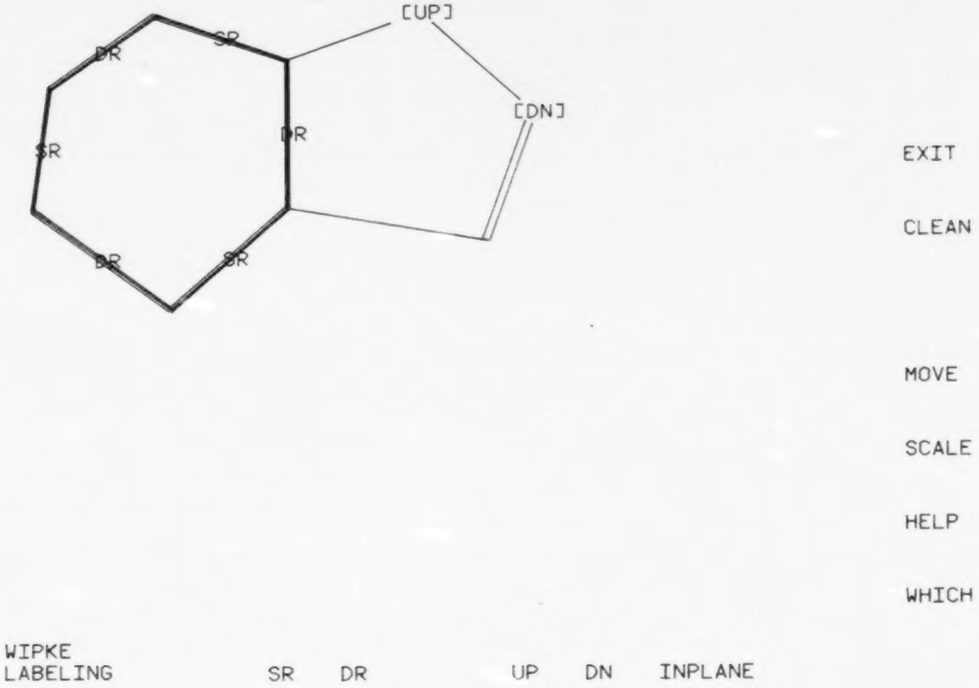


Figure 2-8. Wipke quiz.

You need to supply two kinds of additional information through this quiz. First, the energy minimization procedure has a different convention for resonant bonds (as in benzene) from the MAKE MOLECULE command. It expects alternating single resonant (SR) and double resonant (DR) bonds. Thus, if your ring has resonant bonds, you must relabel them by using the SR and DR light buttons. These work like the BOND light button of the MAKE MOLECULE quiz (see above). Second, if the ring is not planar, the energy minimization procedure will probably be more efficient and accurate if you indicate which ring atoms are out of the plane of the ring relative to other atoms. Pointing to the UP or DOWN light buttons and then to the appropriate atoms accomplishes this; the INPLANE light button is used to undo a previous UP or DOWN specification. The MOVE, SCALE, and CLEAN light buttons provide the usual functions, and are helpful in keeping the display understandable. Pointing to the WHICH light button causes the ring(s) of interest to PROPHET to be flashed again. This is important since the program only accepts specifications for these atoms and bonds.

Pointing to the EXIT light button in the Wipke quiz directs the COMPUTE MODEL command to proceed with the energy minimization. This may take a substantial time, up to several minutes of computer time for each ring. PROPHET types [MINIMIZING...DONE] as each ring is computed, then re-enters the Wipke quiz if other rings are to be specified.

The energy minimization procedure uses an iterative algorithm; this usually means that the longer it runs the better the results. Poor results usually manifest themselves as uncommonly long bonds. The time allowed for the energy minimization has been chosen to permit reasonably large ring systems (e.g., a steroid nucleus) to be computed with reasonable accuracy. Occasionally, the minimization fails, either by stopping and typing an explicit error message, or by producing a strange conformation. The energy minimization procedure tries to retain the stereochemistry about each atom. If the minimization procedure changes the conformation about an atom, it considers this an error and types out "ATOM INVERTED". Recomputing the model with different specifications from the Wipke quiz may help. The less restricted the ring, the more likely the minimization will take an undue amount of time or fail altogether.

If you are happy with the result of the Wipke energy minimization procedure, you may wish to use the ADD RING quiz described above to add the computed ring conformation to your PRIVATE or SHARABLE ring dictionary. Once you have done this, it will not be necessary for the Wipke program to recalculate the energy minimization for each molecule containing the same ring structure.

Refining Whole Molecules

Although the Wipke energy minimization program is used primarily to compute ring conformations in the context of the COMPUTE MODEL command, it can also be invoked explicitly on an entire molecule via the command:

```
REFINE MODEL [OF] molecule-name
```

In this case, the existing model coordinates for the atoms of the molecule are used as initial values in the iterative energy minimization program. When you use the REFINE MODEL command, PROPHET asks a number of questions about limitations on the amount of computation to be performed, and inquires whether you wish the computations to be performed by a batch job.

Other Energy Minimization Tools

Within PROPHET, there is at present one other energy minimization program in addition to the Wipke program. This program, called QCFF, was developed in 1974 by Huber, Sharon and Warshel. It can be accessed through the public procedures EDITMODEL and REFINEMOL, which are described in Chapter V of this manual.

Setting Parameters of the Model

In many cases you may have specific structural information about a molecule -- such as bond lengths, bond angles, or dihedral angles -- that you would like to use in computing the molecule's conformation. You can inform PROPHET of this information by setting up some <u>constraints</u>. These constraints are then used when you issue the COMPUTE MODEL command, as long as they apply to atoms and bonds outside of ring systems.

There are five basic types of constraints that may be specified: bond lengths, dihedral angles, bond directions, bond angles and chirality. Detailed descriptions of how you can specify each of these types are presented in the following paragraphs.

The Wipke energy minimization program does not take any constraints you may have specified into account (except for certain chirality constraints as described below), nor does the use of pre-existing ring conformations in a ring dictionary. If you do specify some constraints within ring systems, you will be informed that these constraints were ignored when COMPUTE MODEL completes the calculation of the molecule's conformation.

Bond Length Constraints

Since some compounds have non-standard bond lengths, you may sometimes wish to specify one or more of these before a model is computed. This is accomplished using the command

```
                                                      value
CONSTRAIN [BOND] LENGTH [OF] bond-indication [TO]
                                                      FREE
```

where "value" is the desired length of the bond in angstroms. Specifying FREE instead removes any existing bond length constraint. The bond of interest can be indicated by pointing to it in a display of the molecule, or you can type:

```
BOND bond-index [OF] molecule-name
```

For example, a typical command would be:

```
CONSTRAIN BOND LENGTH OF BOND 3 OF ACETONE TO 1.25
```

A bond length constraint has no effect if it is applied to a bond in a ring. Bond length constraints do not take effect until the next time the model of the molecule is computed.

Dihedral Angle Constraints

You can specify the dihedral angle (or torsion angle) between three connected bonds with the command:

```
                                                  angle
                                                  CIS
                                                  TRANS
CONSTRAIN DIHEDRAL [ANGLE] bond-triple [TO]       GAUCHE
                                                  GAUCHEP
                                                  GAUCHEM
                                                  FREE
```

The "bond-triple" can consist of pointing to the three bonds, if the molecule is currently displayed. If not, you can type:

```
bond-index [,] bond-index [,] bond-index [OF] molecule-name
```

The dihedral angle is defined to be the clockwise torsion angle between the first and third specified bonds when looking down the axis of the second bond toward the third bond. Figure 2-9 is a Newman projection looking down the axis of the second bond.

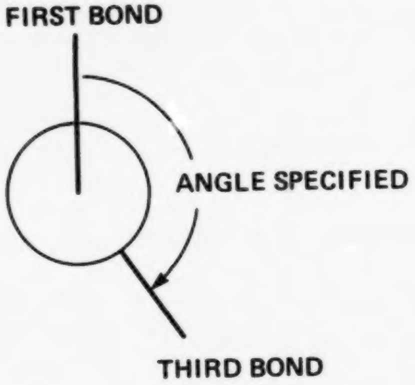


Figure 2-9. Newman projection of the three adjacent bonds demonstrating the angle specification for a dihedral angle constraint.

The dihedral angle can be set to a number of degrees ("angle"), or you can use one of the following angle names:

CIS	0.0 degrees
TRANS	+180.0
GAUCHE	+60.0
GAUCHEP	+60.0
GAUCHEM	-60.0

Using the word FREE instead of an angle or angle name removes any existing dihedral angle constraint applying to the three specified bonds. An example of a typical dihedral angle command is as follows:

```
CONSTRAIN DIHEDRAL ANGLE BONDS 1,2,4 OF ACETONE TO GAUCHE
```

As with bond length constraints, a dihedral angle constraint cannot be imposed on the bonds of a ring. Also, the constraint does not take effect until the next time a model is computed.

Direction Constraints

You can assign relative directions to each bond adjacent to a particular atom in the molecule with the following command:

```
CONSTRAIN DIRECTIONS [AT] atom-spec

bond-specs TO dir-specs<GO>
```

The atom-spec indicates the central atom about which bond directions are to be constrained. You can either point to the atom in a display of the molecule or type:

```
                   [IN]
[ATOM] atom-index       molecule-name
                   [OF]
```

The bond-specs specify each of the bonds surrounding the atom; the bonds can be listed explicitly, or by indicating the atoms (other than the central atom) to which they are attached. In all, there are four forms for the bonds-spec:

1. Typing BONDS and then pointing to each of the bonds with the tablet pen.

2. Typing BONDS, followed by a list of the numeric indices of each of the bonds, separated by commas.

3. Typing ATOMS and then pointing to each of the atoms surrounding the central atom.

4. Typing ATOMS, followed by a list of the numeric indices of each of the atoms surrounding the central atom, separated by commas.

The dir-specs indicate the relative vector directions to be assigned to each of the bonds surrounding the central atom (the first direction is assigned to the first bond mentioned in the bond-specs, etc). The dir-specs can have one of two forms:

1. The name of a variable or a constant MATRIX (FLOAT ARRAY ARRAY), whose elements (rows) will be interpreted as direction vectors.

2. A list of VECTOR (FLOAT ARRAY) variables or constants, separated by commas.

The VECTORs or rows of the MATRIX must have three elements each (X, Y, Z), and not all the elements can be 0. These vectors are interpreted as directions in an arbitrary three-dimensional space. When a model of the compound is computed (using the COMPUTE MODEL command), the specified bonds are assigned those specified relative directions and then rotated to make the bonds fit the rest of the molecule.

A DIRECTIONS constraint is automatically overridden by any other CONSTRAIN DIRECTIONS command (or CONSTRAIN BOND ANGLES command, as discussed below) which mentions the same central atom. Also, a DIRECTIONS constraint can be removed with the command:

```
CONSTRAIN DIRECTIONS [AT] atom-specs [TO] FREE<GO>
```

A typical example of a CONSTRAIN DIRECTIONS command is as follows:

```
CONSTRAIN DIRECTIONS AT ATOM 2 OF ACETONE
BONDS 1,2,3 TO [[-1.,0.,0.],[.33381,.94264,0.],
[.55919,-.82904,0.]]
```

More detailed examples are given below.

Bond Angle Constraints

You can assign specified inter-bond angles among the set of bonds adjacent to a particular atom in the molecule with the following command:

```
CONSTRAIN [BOND] ANGLES [AT] atom-spec bond-specs
[TO] angle-specs<GO>
```

The atom-specs and bond-specs have the same form and effect as in the CONSTRAIN DIRECTIONS command described above; they specify the central atom and list the bonds surrounding it.

The angle-specs indicate the angles in degrees among all the bonds surrounding the central atom. They have one of two forms:

1. A list of numeric variables or constants, separated by commas.
2. A FLOAT ARRAY (or VECTOR) whose elements are interpreted as the list of angles.

The relationship between the order of the bonds and the angles is somewhat complicated, and has different forms depending on the number of bonds surrounding the central atom:

1. [1 bond]: In this trivial case, no angles are necessary.
2. [2 bonds]: In this case, the angle-specs consist of only one angle, that formed by the two specified bonds (the order is unimportant).
3. [3 bonds]: Suppose the bonds in the bond-specs are B1,B2 and B3. Then the angle-specs list three angles: A12, A23, and A31 (where Aij is the angle between Bi and Bj), as shown in Figure 2-10. Notice that three angles alone do not completely specify the directions of the bonds: there is still the question of chirality or handedness. For this reason, it is assumed that bonds B1, B2, and B3 form a "right-handed triple" (analogous to the X, Y and Z axes); by changing the order of the bond-specs, this degree of freedom can be controlled. Notice that the sum of the three angles cannot be greater than 360 (and when their

sum is exactly 360, the three bonds are coplanar). Another more subtle requirement is that no bond exceed the sum of the other two (the "triangle inequality").

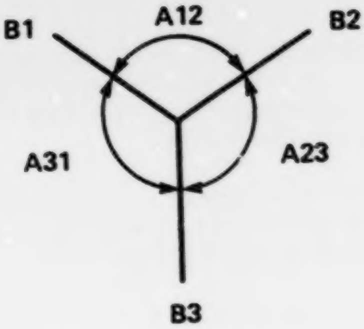


Figure 2-10. Relationship of angles A12, A23 and A31 to bonds B1, B2, and B3 about a central atom with three bonds.

4. [4 bonds]: Let us refer to the bonds in the bond-specs as B1 through B4. Then the angles of the angle-specs are A12, A23, A31, A14, A24, and A34, as shown in Figure 2-11.

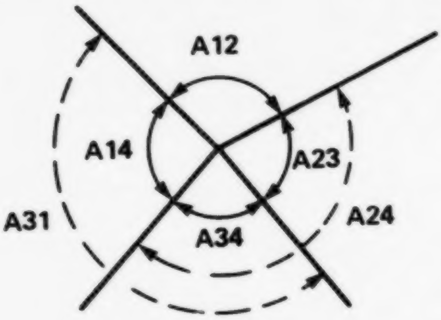


Figure 2-11. Relationship of angles A12, A23, A24, A14, A24 and A34 to bonds B1, B2, B3 and B4 about a central atom with four bonds.

The first three angles work as for the 3-bond case above. When angles A14 and A24 are added, everything except the chirality is specified. The last angle, A34, is somewhat redundant, since its magnitude has already been determined by the other five angles. Since the resulting system is somewhat overconstrained, the angles must be specified rather accurately or there will be a geometric inconsistency.

5. [5 or more bonds]: These cases are not currently handled, and CONSTRAIN DIRECTIONS must be used instead.

When a BOND ANGLES constraint is imposed, PROPHET checks to be sure that all and only those bonds surrounding the central atom have been specified, and that the set of angles specified is geometrically consistent. If this is the case, the information is placed in a row of the molecule's CONSTRAINTS table.

A BOND ANGLES constraint is overridden by any subsequent BOND ANGLES or DIRECTIONS constraint which mentions the same central atom. As with the DIRECTIONS constraints, a BOND ANGLES constraint can be removed with the command:

```
CONSTRAIN BOND ANGLES [AT] atom-specs [TO] FREE<GO>
```

An example of a typical CONSTRAIN BOND ANGLES command is as follows:

```
CONSTRAIN BOND ANGLES AT ATOM 2 OF ACETONE
BONDS 1,2,3 TO 109.5,125.25,125.25
```

More detailed examples are provided below.

Bond Angle and Directions Reminders

Here are several points to remember:

1. No more than four bonds can be constrained using BOND ANGLES constraints; for greater numbers of bonds, use the CONSTRAIN DIRECTIONS command.

2. BOND ANGLES and DIRECTIONS contraints are mutually exclusive: only one can apply to a particular central atom at any time. The last BOND ANGLES or DIRECTIONS constraint set for a prticular central atom is the one that COMPUTE MODEL uses.

3. The bond-specs in a CONSTRAIN DIRECTIONS or CONSTRAIN BOND ANGLES command must contain all the bonds (or attached atoms) surrounding the central atom and no others. The relationship of their order to the order of the angles or directions specified is discussed above.

4. PROPHET does not allow you to set a CHIRALITY constraint (see below) for a central atom which has a BOND ANGLES or DIRECTIONS constraint.

5. As with most other PROPHET molecule constraint types, a BOND ANGLES or DIRECTIONS constraint is ignored when applied to a central atom which is part of a ring system.

Chirality Constraints

Finally, you can specify the chirality about a stereo center using the command:

```
                                                     CC
CONSTRAIN CHIRALITY [AT] atom-spec bond-specs [TO]   CW
                                                     FREE
```

You can specify the chiral "center" atom (the atom-spec) by pointing to it in a display of the molecule or by typing:

```
ATOM atom-index [OF] molecule-name
```

You then specify the substituents (bond-specs) which are used to define the chirality. You can do this by pointing to three atoms or three bonds connected to the center atom or by typing:

```
BONDS bond-index [,] bond-index [,] bond-index
```

or

```
ATOMS atom-index [,] atom-index [,] atom-index
```

Finally, you specify the order of circulation of the three substituents, as if you were looking at them from the position of the fourth substituent (see Figure 2-12): CC stands for counterclockwise and CW for clockwise. You can also use the word FREE to remove any existing chirality constraint about the center atom.

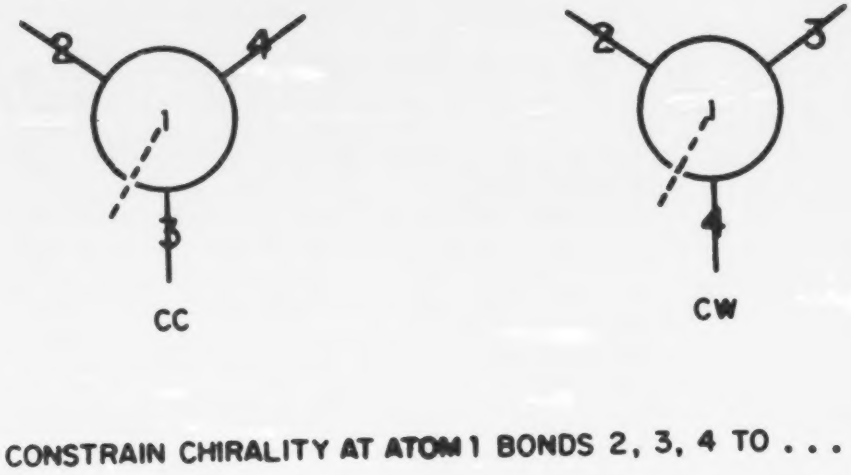


Figure 2-12. Illustration of meaning of CC and CW chirality constraints.

PROPHET CHIRALITY constraints work properly only for center atoms having exactly four bonds. Unlike other types of constraints, chirality constraints can be applied to atoms in rings, provided that at least two substituents of the ring atom are not also in the ring. Like the other constraints, however, the CHIRALITY constraint does not go into effect until a model of the molecule is computed.

Examining Constraints

You can examine the constraints that you have imposed on a molecule with the command:

```
DISPLAY CONSTRAINTS OF molecule-name
```

The command displays a table in which each row describes a single constraint in terms of its type, which atoms and/or bonds are constrained, and the value they are constrained to.

Constraint Examples

1. To constrain bonds 1, 2, and 3 surrounding atom 1 of molecule M to lie in a planar trigonal arrangement, one could type:

   ```
   CONSTRAIN DIRECTIONS AT ATOM 1 OF M BONDS 1,2,3 TO
   [1.,0.,0.], [-.866,0.5,0.],[-.866,-0.5,0.]<GO>
   ```

   Note the use of .866 (cosine of 120 degrees) and 0.5 (sin(120)) to place the three vectors 120 degrees apart in the XY plane; the same effect could be achieved with a BOND ANGLES constraint.

2. Suppose we wish to constrain bonds connected to atoms 1 through 6 to lie in an octahedral arrangement about atom A. We could use the same approach as above, typing in six vector constants, but it is easier to exploit some of the vector and matrix arithmetic capabilities of PROPHET (see Chapter IV):

   ```
   DECLARE (X,Y,Z) VECTOR<GO>
   X=[1.,0.,0.]<GO>
   Y=[0.,1.,0.]<GO>
   Z=[0.,0.,1.]<GO>
   CONSTRAIN DIRECTIONS AT ATOM 1 OF M
   BONDS 1,2,3,4,5,6 TO -Z,X,Y,-X,-Y,Z<GO>
   ```

3. To constrain bonds 1, 2 and 3 surrounding atom 1 of molecule M to lie in a planar trigonal arrangement, one could type:

   ```
   CONSTRAIN BOND ANGLES AT ATOM 1 OF M BONDS 1,2,3
   TO 120.,120.,120.<GO>
   ```

4. To constrain bonds B1 through B4 surrounding atom A of molecule M to point in the +X, +Y, -X, and -Y directions, type:

   ```
   CONSTRAIN BOND ANGLES AT ATOM A OF M BONDS
   B1,B2,B3,B4 TO 90.,90.,180.,90.,180.,90.<GO>
   ```

Figure 2-13 illustrates this example.

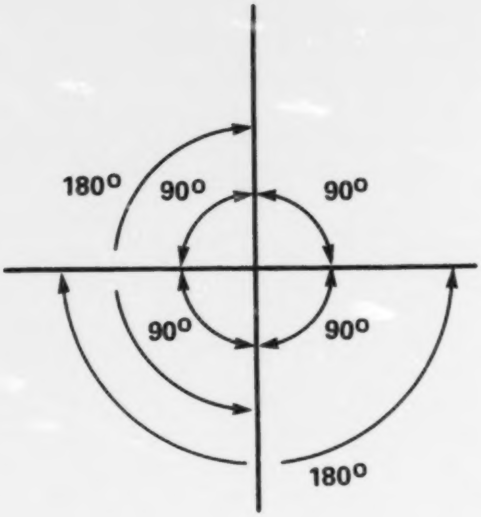


Figure 2-13. Constraint example 4.

5. Finally, to constrain bonds B1 through B4 to have the usual sp3 hybridization, and to have B1, B2, and B3 circulate counterclockwise when viewed from B4, one would type:

   ```
   CONSTRAIN BOND ANGLES AT ATOM A OF M BONDS B1,B3,B2,B4
   TO 109.47,109.47,109.47,109.47,109.47,109.47<GO>
   ```

   Note that the chirality requirement forced us to specify the first three bonds in the order B1, B3, and B2, since it is in that order that they would form a right-handed triple when arranged as specified.

Part 3: Displaying a Molecule

Many of the basic PROPHET display commands that work for TABLEs and GRAPHs, such as DISPLAY, EAD and PRINT, are also used for MOLECULES. The simplest form of these commands is:

```
EAD                          SKETCH
DISPLAY molecule-name THREED
PRINT                        STEREO
```

where the standard default is SKETCH. As usual, the EAD command erases the screen before the molecule is drawn with its upper left-hand corner a few lines below the upper left-hand corner of the screen, while DISPLAY causes the molecule to be displayed in whatever space is still available on the screen at the time you issue the command. The PRINT command erases the screen, draws the molecule and activates the hard copier. The screen is then erased again and whatever displays were present on it before the command was typed are restored.

With SKETCH, this command gives you the sketch of your molecule, essentially the same way you drew it in using MAKE MOLECULE. When a molecule is displayed, all non-carbon atoms are labeled with their chemical symbols, multiple bonds are shown with multiple lines, and resonant bonds are shown as heavy lines. System-added hydrogens are not displayed, but any hydrogens you explicitly specified when defining the molecule are displayed.

Displaying the Model

If a model exists for your molecule, you may display it as a three-dimensional structure using the display option THREED, which will produce a two-dimensional projection of the three-dimensional coordinates stored in the model. No hydrogens are displayed, neither those you specified nor those added by the system. The option STEREO will yield two images of the THREED view in slightly different orientations. By viewing them with a stereo viewer (or by relaxing your eyes), you can get a three-dimensional effect.

Examples

Taking our sample molecule named ACETONE, the command

```
EAD ACETONE
```

produces the display shown in Figure 2-14.

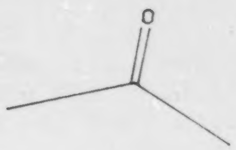


Figure 2-14. Display of ACETONE SKETCH.

If you have computed a model for your molecule, the command:

```
EAD ACETONE THREED
```

produces the display shown in Figure 2-15.

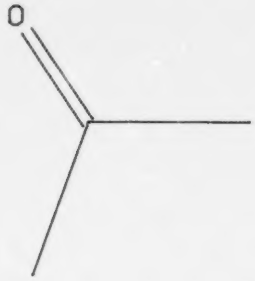


Figure 2-15. Display of ACETONE THREED.

Other Display Options

Several other optional qualifiers can be appended to the DISPLAY command for molecules (standard defaults are underlined):

NOATOMINDEX	NOBONDINDEX	NODREIDING	NOCOMPLETE	TYPES
ATOMINDEX	BONDINDEX	DREIDING	COMPLETE	NOTYPES

ATOMINDEX causes the numeric index of each atom to be displayed beside the atom. BONDINDEX causes the numeric index of each bond to be displayed beside the bond. DREIDING causes all of the atom type labels to be suppressed (carbon labels are normally suppressed anyway), giving the display the stick-figure appearance of a Dreiding model. Finally, the COMPLETE option causes the hydrogen atoms of a molecule to appear when the molecule is displayed THREED or STEREO; normally they are suppressed. Notice that this option does not apply when the molecule is displayed as a SKETCH. There, user-supplied hydrogens are shown while system-added hydrogens are not shown. It is not possible to see system-added hydrogens in a SKETCH because they have no sketch coordinates. TYPES has the same effect as NODREIDING and NOTYPES has the same effect as DREIDING. Do not use both DREIDING and TYPES qualifiers in the same command.

Setting Default Options

The DEFAULT command for displaying molecules takes the following form:

```
DEFAULT MOLECULES [TO display-qualifiers]
```

Any of the display qualifiers described above may be set up as default options by including them in the DEFAULT command just as they would appear in the DISPLAY command. For example, the command

```
DEFAULT MOLECULES TO THREED COMPLETE
```

would cause all subsequent unqualified molecule DISPLAY commands to display the three-dimensional molecular model complete with all hydrogens. Typing the simple form of this command:

```
DEFAULT MOLECULES<GO>
```

produces a display of the qualifiers that are currently set as the default options and gives you the chance to change any of them that you wish. In Figure 2-16, we have used this command to change the molecule display defaults to THREED COMPLETE ATOMINDEX and then displayed the ACETONE molecule.

```
CURRENT DEFAULTS FOR DISPLAYING MOLECULES ARE SHOWN BELOW.

TO LEAVE THE DEFAULTS AS THEY ARE, JUST TYPE <GO>.

TO RETURN THE DEFAULTS TO THE SYSTEM STANDARDS, JUST
TYPE "STANDARD" OR "S"

TO CHANGE ANY OF THE DEFAULTS, TYPE THE NUMBERS OF THE
DEFAULTS YOU WANT TO CHANGE, SEPARATED BY A COMMA OR SPACE.
?1 2 4<

NEW DISPLAY MODE (SK(ETCH), 3D, ST(EREO))?3D< >DIS ACETONE<
```

1	SKETCH
2	NOCOMPLETE
3	NODREIDING
4	NOATOMINDEX
5	NOBONDINDEX
6	TYPES

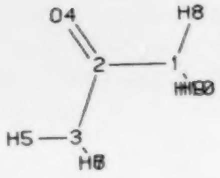


Figure 2-16. Sample use of DEFAULT MOLECULES quiz.

Selective Displays

Sometimes you may want to display only certain parts of a molecule, say the ring nucleus of a steroid, or some sub-sequence of residues in a protein. To do this, you can use the <u>selective</u> <u>display</u> option for molecules, which involves appending an expression of the form:

```
             ATOMS
SELECTING          [BY] subset-name
             BONDS
```

to any molecule display command.

Before using the display command, you must indicate which atoms and/or bonds you wish to select. One way to do this is to use the MARK command to define a subset of atoms or bonds:

```
      ATOMS
MARK        [OF] molecule-name [FOR] subset-name
      BONDS
```

When you issue the command, PROPHET displays the molecule (unless it is already on the screen) and asks you to point to atoms (or bonds) you wish to select, terminated with a <GO>. The subset-name (a name enclosed in single quotes) can be used subsequently to refer to the set of atoms or bonds you have just selected, as described below. The effect of the MARK command is to add a column with the heading designated by subset-name to the molecule's ATOMS or BONDS table containing information about which atoms or bonds were marked. Chapter III of this manual explains the ATOMS and BONDS tables and describes more advanced techniques for defining subsets of atoms or bonds.

You can also use the MARK command to add more atoms or bonds to a subset previously selected using MARK. You can remove atoms or bonds from a subset using the command:

```
        ATOMS
UNMARK         [OF] molecule-name [IN] subset-name
        BONDS
```

When you have created a subset of selected atoms and/or bonds, you can display these using the SELECTING option appended to any molecule display command:

```
           ATOMS
SELECTING          [BY] subset-name
           BONDS
```

You can select either ATOMS or BONDS, or both, by using two SELECTING phrases. If both atoms and bonds are selected then just the atoms and bonds in the subsets are displayed. If you use only the SELECTING ATOMS option, then the atoms you selected are shown, together with any bonds which connect two selected atoms. Similarly, if you use only the SELECTING BONDS option, then the bonds you selected and any atoms connected to them are displayed.

An Example

Suppose you have a simple molecule called MOL1 that appears as illustrated in Figure 2-17 when you type the command:

```
EAD MOL1 THREED COMPLETE<GO>
```

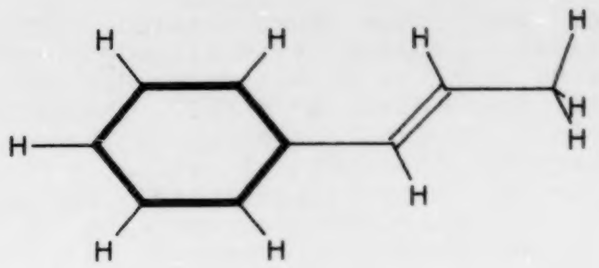


Figure 2-17. Example molecule MOL1 displayed THREED COMPLETE.

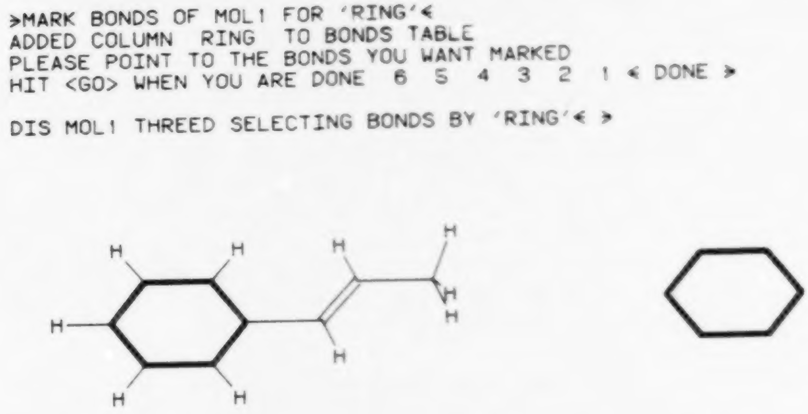
You could create a display of only the ring portion of MOL1 by typing the command:

```
MARK BONDS OF MOL1 FOR 'RING'<GO>
```

while MOL1 is displayed on the screen, then pointing to the six resonant bonds in the benzene ring, and finally typing the command:

```
DISPLAY MOL1 THREED SELECTING BONDS BY 'RING'<GO>
```

The screen would appear as illustrated in Figure 2-18.



```
>MARK BONDS OF MOL1 FOR 'RING'<
ADDED COLUMN  RING  TO BONDS TABLE
PLEASE POINT TO THE BONDS YOU WANT MARKED
HIT <GO> WHEN YOU ARE DONE  6  5  4  3  2  1 < DONE >

DIS MOL1 THREED SELECTING BONDS BY 'RING'< >
```

Figure 2-18. Display screen after marking the ring bonds in MOL1 and selectively displaying them.

As a further example, you could take the same molecule and display it again with the command:

```
EAD MOL1 THREED COMPLETE<GO>
```

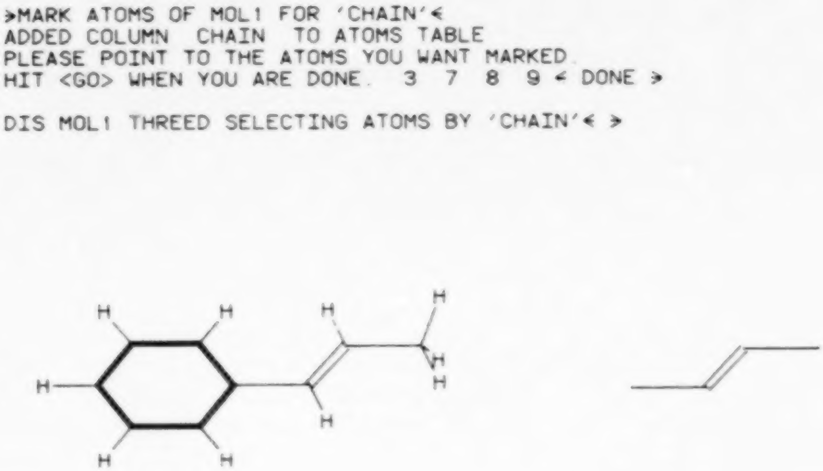
Then type the command:

```
MARK ATOMS OF MOL1 FOR 'CHAIN'<GO>
```

After you have pointed to the atoms in the side chain, type:

```
DISPLAY MOL1 SELECTING ATOMS BY 'CHAIN'<GO>
```

At this point, the screen appears as illustrated in Figure 2-19.



```
>MARK ATOMS OF MOL1 FOR 'CHAIN'<
ADDED COLUMN  CHAIN  TO ATOMS TABLE
PLEASE POINT TO THE ATOMS YOU WANT MARKED
HIT <GO> WHEN YOU ARE DONE.  3  7  8  9 < DONE >

DIS MOL1 THREED SELECTING ATOMS BY 'CHAIN'< >
```

Figure 2-19. Display screen after marking the side chain atoms in MOL1 and selectively displaying them.

Labeled Displays

In molecule displays all atoms (except carbons) are normally labeled with their atom type; bonds are usually not labeled, except in the MAKE MOLECULE quiz. You can display atom indices with the atom types using the ATOMINDEX qualifier and label bonds with bond indices using the BONDINDEX qualifier, as described above. A more advanced option allows you to define arbitrary labels for atoms or bonds; you can use this option by adding the following phrase to any molecule display command:

```
                 ATOMS
        LABELING        [FROM][COLUMN] label-col
                 BONDS
```

The "label-col," which may be either a FIXED column number or a TEXT column heading, refers to a column in the ATOMS or BONDS tables. These tables are part of the molecule data structure as described in Part 1 of Chapter III. This column contains additional labels to be applied to the atoms or bonds. The labels contained in column label-col are appended to whatever other labels are already being displayed, such as the atom types. If you wish to label atoms only with the labels in label-col, you should use the NOTYPES display qualifier. For example, the command:

```
EAD MOL1 THREED NOTYPES LABELING ATOMS FROM COL 8
```

would display MOL1 without atom types but with labels taken from column 8 of the ATOMS table. There is currently no mechanism analogous to the MARK command for setting up label columns. Consult Chapter III of this manual for details of how this can be done.

Changing the Molecule Display

Once you have displayed a molecule, you may use a number of display-handling commands to alter its appearance. For example, you may use the SCALE command to change the size of the display and the MOVE command to translate it. There are also a number of display manipulation commands which are particularly suited to three-dimensional molecule displays.

Scaling the Molecule Display

You may change the size of a molecule display with the usual PROPHET SCALE command:

```
SCALE [scale-factor] [point-to-scale-center]
```

Just typing SCALE<GO> results in scaling the last molecule currently displayed on the screen by the default scale factor of 0.5 with the scaling centered on the upper left hand corner of the display. You may specify any other relative scale factor you wish by typing a number after the word SCALE, and you may specify a different center for the scaling operation by pointing to the location you wish to center upon to terminate the command. Note that PROPHET does not allow you to apply different scale factors along different axes for a molecule display; all scale factors are applied uniformly along the X, Y and Z axes. Any time a molecular model is displayed THREED or STEREO with a scale factor other than 1.0, the scale factor that is currently in effect is shown at the top of the display.

Moving the Molecule Display

You can use the usual MOVE command to change the location of a molecule display on the screen:

```
MOVE point point
```

After typing MOVE, point on the data tablet to an atom or bond in the molecule you wish to move, and then point to the location on the screen where you wish to have that atom or bond located. The screen is erased and the molecule redrawn with the atom or bond you indicated located at the position you pointed to.

Turning the Molecule Display

You can rotate a molecule SKETCH display with the TURN command:

```
TURN angle [AROUND] Z [point]
```

and you can rotate a molecule THREED or STEREO display with the more general form of the TURN command:

```
                       X [point]
TURN angle [AROUND]    Y [point]
                       Z [point]
                       point point
```

The "angle" is the number of degrees to rotate, and the rest of the command specifies an axis about which to rotate (clockwise, looking in the direction of the axis away from the origin).

The X-axis crosses the screen from left to right, with right being the positive direction. The Y-axis goes from the bottom to the top of the screen, with the top being positive. Finally, the Z-axis points out of the screen toward you. Note that since the positive direction of the Z-axis is considered to be toward you, rotating the display clockwise around the Z-axis results in an apparent counterclockwise rotation from your vantage point. You may optionally point to some point in the molecule after specifying X, Y, or Z, in which case the rotation occurs around that point.

Instead of typing X, Y or Z in the command, you can use the data tablet to indicate two points in the molecule (atoms, bonds, or just points). In that case, the two indicated points form the axis about which the display is rotated, with the positive direction going from the first point toward the second. If the display of the molecule is THREED or STEREO, the rotation occurs in three dimensions. If the display is SKETCH, the rotation takes place in two dimensions, and you must specify Z as the axis of rotation.

Orienting the Molecule Display

An alternative way to rotate the display is to specify the desired orientation which the molecule should have after rotation, using the command:

```
                     atom-spec atom-spec
ORIENT [ALONG] axis  bond-specification
                     point point
```

In this command, you first specify the desired orientation in three-dimensional space, then a direction in the molecule that is to be rotated to achieve the desired orientation. The "axis" can be specified by typing X, Y or Z (meaning, as above, the directions relative to the display screen), or by pointing to two points in the display. The remaining items specify what is to be oriented: you can point to a pair of atoms, or type:

```
ATOMS atom-index atom-index
```

Alternatively, you can specify a bond, either by pointing to it or by typing:

```
BOND bond-index
```

Finally, you can just point to any two points on the display; in the latter case, the display is turned so that these points face in the direction of the axis defined.

The actual rotation that results from the ORIENT command is the "minimal" rotation needed to take the specified atoms, bonds, or points to the desired orientation. Unfortunately, this may confound the effects of a previous ORIENT command. Sometimes, you can use a combination of ORIENT and TURN commands to achieve a desired effect.

ORIENT and TURN Example

Suppose, for example, you wish to rotate the display of a molecule so that atoms 4 and 5 lie along the Y-axis and atom 8 (as well as 4 and 5) lies in the YZ-plane. Figure 2-20 shows the display that results from the command:

```
EAD ACETONE THREED COMPLETE NOTYPES ATOMINDEX<GO>
```

If you now type the command:

```
ORIENT ALONG Y ATOMS 4 5<GO>
```

the resulting display is that shown in Figure 2-21.

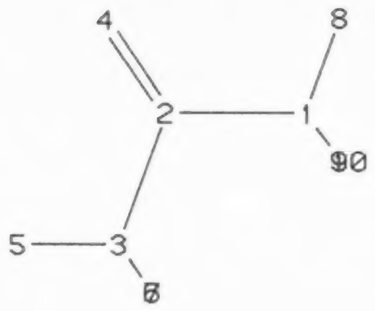


Figure 2-20. ACETONE displayed THREED COMPLETE NOTYPES ATOMINDEX.

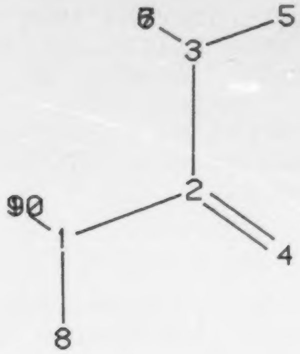


Figure 2-21. ACETONE display after orienting atoms 4 and 5 along the Y-axis.

Now all that is needed to bring atom 8 into the YZ-plane is to type a series of commands of the form:

```
TURN angle AROUND Y
```

until atom 8 lines up vertically with atoms 4 and 5. In the case of our ACETONE illustration, the command that does the trick is:

```
TURN 9Ø AROUND Y<GO>
```

and the resulting display is illustrated in Figure 2-22.

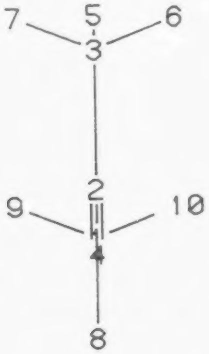


Figure 2-22. ACETONE molecule displayed with atoms 4 and 5 oriented along Y-axis and atom 8 in YZ-plane.

Saving the Appearance of the Display

After manipulating the display of a molecule with the SCALE, MOVE, TURN, and ORIENT commands, you may wish to preserve the state of the display for subsequent sessions. To do this, use the command:

```
FREEZE [point-to-molecule]
```

This saves the state of the display for the indicated molecule (or for the last molecule displayed), so that it will have the same size and orientation when you issue subsequent DISPLAY commands. FREEZE does not alter the stored 3-D model coordinates of the atoms in the molecule; however, if you FREEZE a SKETCH display, the 2-D sketch coordinates stored in the molecule are changed.

Permanent Changes to the Model

All the display manipulations mentioned above (MOVE, SCALE, TURN, ORIENT, and FREEZE) simply affect the appearance of the molecule when it is displayed, but have no effect on the 3-D coordinates of atoms stored in the molecule. There are, however, three display manipulation commands which do affect those stored model coordinates, and it is also possible to explicitly modify the model coordinates for particular atoms as described below.

Twisting a Molecule

You can alter the conformation of a molecule by rotating about a single bond, using the command:

```
TWIST angle [AROUND] atom1 atom2
```

You specify the amount of twist as an "angle" in degrees, and indicate where to twist by pointing to two atoms, or by typing their numeric indices. The effect is to rotate the part of the molecule connected to the second atom by "angle" degrees in the clockwise direction, when viewed looking from atom1 toward atom2. The TWIST command aborts if the specified atoms are not connected by a single, non-ring bond. Again, note that the TWIST command causes a permanent change in the stored atom coordinates of the molecule; this happens whether or not you issue a FREEZE command.

Saving Changed Model Coordinates

A second way of permanently changing the molecule's model is with the command:

```
DEEPFREEZE [point-to-molecule]
```

This command alters the stored model coordinates so that they correspond to the currently displayed orientation. That is, the effects of all TURN and ORIENT commands so far are applied to the stored coordinates. The effects of previous SCALE and MOVE commands are not preserved, however.

Changing the Coordinate Center

You can alter the coordinate center for the molecule's model with the command:

```
                 atom-specification
     CENTER [ON]
                 bond-specification
```

You can point to a particular atom or bond, or type the numeric index of the atom or bond (optionally followed by pointing to the molecule of interest). The effect of the command is to change the stored atom coordinates so that the specified atom, or the center of the specified bond, lies at the origin (coordinates [0 0 0]). Thus, by using a combination of the TURN, ORIENT, DEEPFREEZE and CENTER commands, you can effectively move the model coordinates of a molecule arbitrarily in 3-dimensional space.

Accessing Components of a Molecule

You can use the computed model of a molecule to obtain geometrical information and to access various pieces of the molecule. Each of the expressions described below acts as a function and returns a value. You can TYPE or DISPLAY the value, or you can SET a variable to it. The first expression may be used to determine the number of atoms, bonds, or constraints in a molecule:

```
                  ATOMS
NUMBER [OF]       BONDS      [IN] molecule-name
               CONSTRAINTS
```

This expression returns a value of type FIXED. This is most useful in a procedure, where you want to cycle through all the atoms, bonds, or constraints of a molecule to carry out some operation on each of them. For example, you could use the expression as the upper limit of a DO statement,

```
DO I=1 TO NUMBER OF ATOMS IN ACETONE;
```

See Chapter III for more details about molecule-oriented programming in PL/PROPHET.

Atomic Coordinates from a Model

It is possible to use expressions to obtain information about the locations of various atoms in the molecule after a model has been computed. For example, the expression:

```
XCOORDINATE
YCOORDINATE [OF] ATOM atom-index [OF] molecule-name
ZCOORDINATE
```

returns a FLOAT which is the X-, Y- or Z-coordinate, in angstroms, of the atom referenced relative to an arbitrary center of coordinates. If you have used the CENTER command (see above), the center of the coordinate system is the atom or bond you chose. In addition to obtaining each of the three coordinates separately, you can obtain them all as a single three-element array (VECTOR) by using the expression:

```
COORDINATES [OF] ATOM atom-index [OF] molecule-name
```

This expression returns a 3-element FLOAT array (VECTOR). Chapter IV of this manual describes the vector and matrix facilities of the PL/PROPHET programming language, which may be helpful in working with atomic coordinates. It is possible, however, to TYPE or DISPLAY coordinates of an atom without declaring any variables.

Distances in a Model

In addition to the coordinates of the various atoms in a molecule model, it is possible to obtain directly the distances between two atoms. This is accomplished using the expression:

```
                 point-to-atom point-to-atom
DISTANCE [OF]
                 ATOMS atom1 [,] atom2 [OF] molecule-name
```

where "point-to-atom" means pointing to the atom intended, and "atom1" and "atom2" are numeric indices of atoms (the latter may be used when the molecule is not currently displayed). This expression returns a FLOAT which is the distance, in angstroms, between the two atoms indicated. When the two atoms are joined by a bond, it is also possible to use the expression:

```
                    point-to-bond
BOND LENGTH [OF]
                    BOND bond-index [OF] molecule-name
```

This expression also returns a FLOAT number of angstroms.

Angles in a Model

It is also possible to obtain angles between bonds in a molecule. The expression:

```
              point-to-atom point-to-atom point-to-atom
ANGLE [OF]
              ATOMS atom1 atom2 atom3 [OF] molecule-name
```

returns a FLOAT which is the angle defined by the three atoms you reference, either by pointing ("point-to-atom") or by typing their numeric indices ("atom1", etc.). The atoms are taken to define a plane and the second atom indicated is taken to be at the apex of the angle.

The angle between bonds is also returned by the expression:

```
                   point-to-bond point-to-bond
BOND ANGLE [OF]
                   BONDS bond1 bond2 [OF] molecule-name
```

As usual, you can point to the bonds of interest or type their numeric indices.

Dihedral Angles

Finally, you can obtain the dihedral angle formed by four atoms or three contiguous bonds in a molecule with an expression of the form:

```
                           point-to-atom point-to-atom point-
                           to-atom point-to-atom

                           ATOMS atom1 atom2 atom3 atom4
                           [OF] molecule-name
DIHEDRAL [ANGLE] [OF]
                           point-to-bond point-to-bond point-
                           to-bond

                           BONDS bond1 bond2 bond3 [OF]
                           molecule-name
```

Once again, you can either point to the atoms or bonds of interest or type their numeric indices. The FLOAT value returned is the torsion angle between bond1 and bond3 measured clockwise when looking down bond2 toward bond3. If you have specified four atoms, the torsion angle is measured clockwise looking down the bond (or line) between atom2 and atom3 toward atom3.

SET Commands

All the above expressions can appear on the right-hand side of a SET command, but only three of them may also appear on the left-hand side of a SET command, i.e., you may set values into them. These are XCOORDINATE, YCOORDINATE and ZCOORDINATE. Thus, you might give the following commands to put the third atom of ACETONE at coordinates [3.2,2.5,10.3]:

```
SET XCOORDINATE OF ATOM 3 OF ACETONE TO 3.2
SET YCOORDINATE OF ATOM 3 OF ACETONE TO 2.5
SET ZCOORDINATE OF ATOM 3 OF ACETONE TO 10.3
```

All the expressions which refer to information about the molecular model can be used in commands such as TYPE, DISPLAY and SET or in any place a variable name might ordinarily appear. If one of these expressions occurs as the last thing in a command, the command must be terminated with the GO key.

The public procedures MAST and MOLEDAT may also be used to measure distances and angles in molecular models. See Chapter V for details of how to use them.

Perspective Displays

An alternative way to display three-dimensional molecular models is to use the system procedure %PERSPECTIVE. %PERSPECTIVE makes use of the FORTRAN program ORTEP2 written by Carroll Johnson of the Oak Ridge National Laboratory. Figure 2-23 shows a display produced by %PERSPECTIVE for our sample ACETONE molecule.

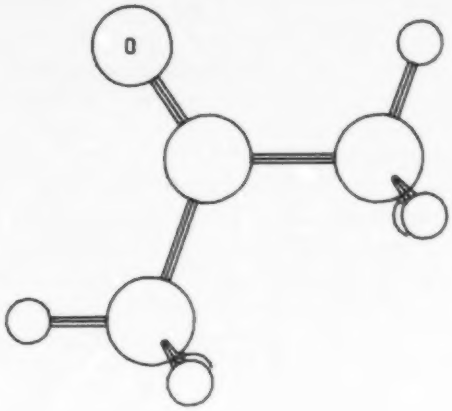


Figure 2-23. %PERSPECTIVE display of ACETONE.

The procedure is used in the following way:

```
CALL %PERSPECTIVE(molecule-name)
```

A model of the molecule must have already been computed so that the procedure can find the atom coordinates needed for this type of display.

The procedure asks a series of questions which allow for some flexibility in the final display.

1. DO YOU WISH TO CHANGE THE DEFAULT SCALE FACTOR OF .5 INCHES/ANGSTROM? If you answer "NO" the display will have the same factor as an unscaled THREED display. If you answer "YES" you are asked for a new scale factor in inches/Angstrom.

2. DO YOU WISH TO CHANGE THE DEFAULT VIEWING DISTANCE OF 30 INCHES? If you answer "YES" the program asks you to type a viewing distance. A shorter distance produces a greater perspective effect (foreshortening).

3. DO YOU WISH TO CENTER ON A SPECIFIC ATOM (THE DEFAULT IS THE "CENTER OF GRAVITY")? If you answer "NO" the display is centered on the screen. If you answer "YES" the program will ask you for the numeric index of the atom to lie at the center of the screen. (Remember to find the index of the center atom before calling %PERSPECTIVE.)

4. DO YOU WISH TO LABEL C AND H ATOMS? Sometimes for speed or display simplicity it is preferred not to label C or H atoms.

5. DO YOU WISH TO DISPLAY H ATOMS? If you answer "NO" neither the H atoms nor any bonds connected to H atoms are displayed, and question 4 pertains only to C atoms.

6. WOULD YOU PREFER TO DISPLAY ATOM INDICES RATHER THAN ATOM TYPES? If you answer "NO" atom symbols are used to label atoms. If you answer "YES" the atom index is used. This is analogous to a THREED ATOMINDEX display.

7. DO YOU WISH TO SPECIFY ATOMIC RADII? If you answer "NO" the procedure begins, after a short pause, to draw a perspective diagram. It uses one third the covalent radius of each atom as the radius of the circle it displays for that atom. If you answer "YES" you are asked to type an atom symbol or "ALL". You are asked for the covalent radius and then if you wish to specify another atom symbol and its covalent radius. If at any point you specify "ALL" as the atom type then the covalent radius that you next specify will be used for all atom types not already specified (i.e., "ALL" is the last radius you can specify).

The usual method of using this procedure is to compute a model of a molecule, display the molecule THREED and manipulate this display until you get an interesting orientation. You then FREEZE the molecule, which changes the molecule's transformer to represent the orientation displayed. Finally you call %PERSPECTIVE with this molecule. This process can be repeated as often as you like.

Immediately after a perspective drawing is complete, the hard copier is automatically activated with the hope of getting a good copy before any noise affects the display. When %PERSPECTIVE has completed the display, it asks you if you want a plot of the display. If you answer "YES" a plot will be made and mailed to you in a few days. This plot will not be affected by any noise you may have had in your original display. Noise can be a serious problem in %PERSPECTIVE displays because these displays require significantly longer drawing times than regular displays and because it is not useful to do a CLEAN, which merely restores the display that was on the screen before you called %PERSPECTIVE. Therefore, if noise occurs, you may wish to call %PERSPECTIVE again.

Quick Perspective Displays

If you wish to get a perspective display of your molecule, but do not want to be bothered with answering all of the questions that %PERPSECTIVE asks, use the quick form of the procedure:

```
CALL %PERSPECTIVEQK(molecule-name)
```

Calling %PERSPECTIVEQK results in the same display you would get if you called %PERSPECTIVE and answered "NO" to questions 1 through 7.

Other Displays

There are two public procedures described in Chapter V that allow you to make other sorts of molecular displays. The public procedure SPACEMOL produces a display of a space-filling model of a molecule containing 3-D coordinates. STEREOBOX produces a stereo display of a molecular model that can be viewed with a special viewing apparatus utilizing polarized filters and a half-silvered mirror. See Chapter V for details of how to use these procedures.

Part 4: Multimolecules

If you are studying the structural interactions of different molecules, such as the docking of an enzyme substrate with the enzyme's active site, you may find the PROPHET multimolecule feature useful. In PROPHET, a MULTIMOLECULE (which may be abbreviated as MULTIMOL or MMOL) is an object consisting of several molecules, all containing 3-D structural information and all related to a common coordinate space. In this chapter, we present commands for constructing and displaying a multimolecule, as well as for manipulating the molecules within a multimolecule relative to each other.

The Multimolecule Quiz

To create a multimolecule, use the command

```
MAKE MULTIMOLECULE unused-name
```

PROPHET creates an empty multimolecule with the specified name, and then places you in the multimolecule quiz, in which you can add, delete, and manipulate the molecules within the multimolecule. The quiz provides you with these functions through a set of command light buttons, as shown in Figure 2-24. As usual, you can indicate a particular light button by pointing to it with the pen on the tablet, or by typing its name. You can also abbreviate command names, where the abbreviation is not ambiguous (PROPHET warns you if it does not recognize a command, or if an abbreviation is ambiguous). To modify an existing multimolecule, type

```
MANIPULATE molecule-name
```

and the same light buttons are displayed.

The light buttons for the multimolecule quiz are displayed in two major groups. The buttons along the right side of the display manipulate the display as a whole and query the multimolecule for information; the light buttons that appear along the bottom of the display screen are used to manipulate individual molecules in the multimolecule. In addition, there is a single light button in the lower right corner, which invokes the HELP command. This command has the form:

```
          [ALL]
     HELP
          command
```

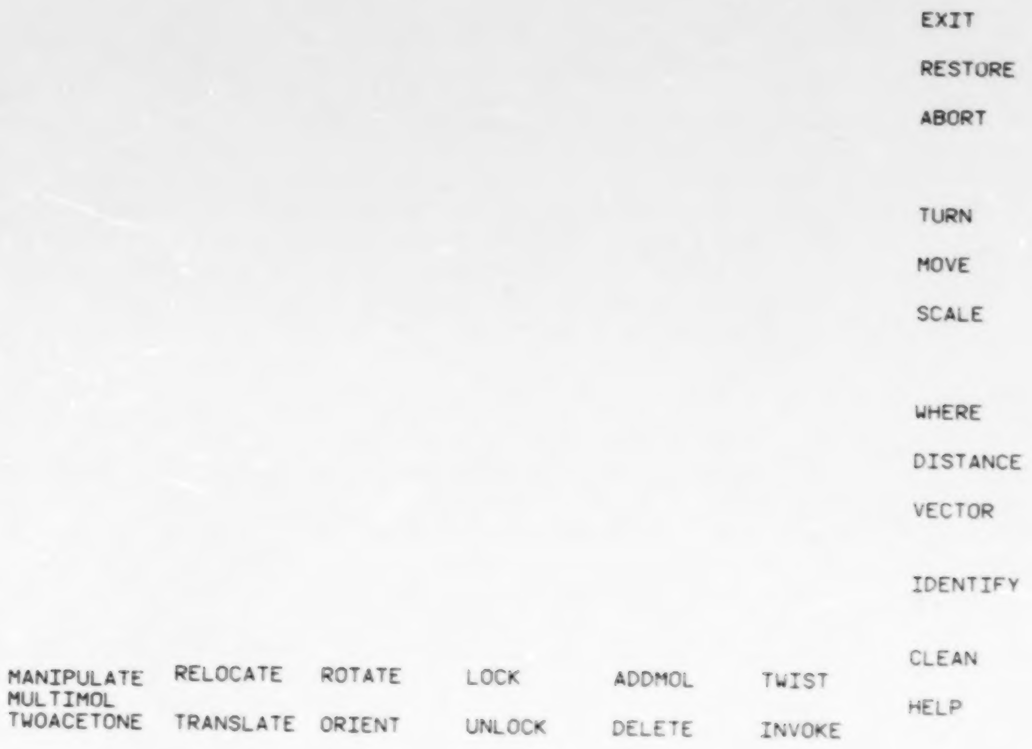

EXIT
RESTORE
ABORT
TURN
MOVE
SCALE
WHERE
DISTANCE
VECTOR
IDENTIFY
CLEAN
HELP
MANIPULATE
MULTIMOL
TWOACETONE
RELOCATE
TRANSLATE
ROTATE
ORIENT
LOCK
UNLOCK
ADDMOL
DELETE
TWIST
INVOKE

Figure 2-24. MAKE/MANIPULATE MULTIMOLECULE quiz light buttons.

In the "ALL" form, which you specify by typing HELP or pointing to the HELP light button and then typing ALL<GO>, PROPHET types out a one-line description of each command in the multimolecule quiz. Use the second form of the command to type a more detailed description of a single command, which you indicate by pointing to its light button, or by typing the name of the command, followed by <GO>.

Many of the commands that follow include the specification "point". This means pointing to an atom, bond, or other point with the pen on the data tablet. When you point to an atom or bond, the multimolecule quiz interprets this as the three-dimensional coordinates of that atom or that bond's midpoint in the multimolecule's reference frame. Pointing to a place that does not coincide with an atom or bond is interpreted as a location in the plane of the display screen, or with the same Z-value as a previously indicated atom or bond.

Adding a Molecule

To add a molecule to the multimolecule, use the ADDMOL command. When you do this, PROPHET types:

```
ENTER NAME OF MOLECULE TO BE ADDED
(OR "MAKE" OR "RESTORE"):
```

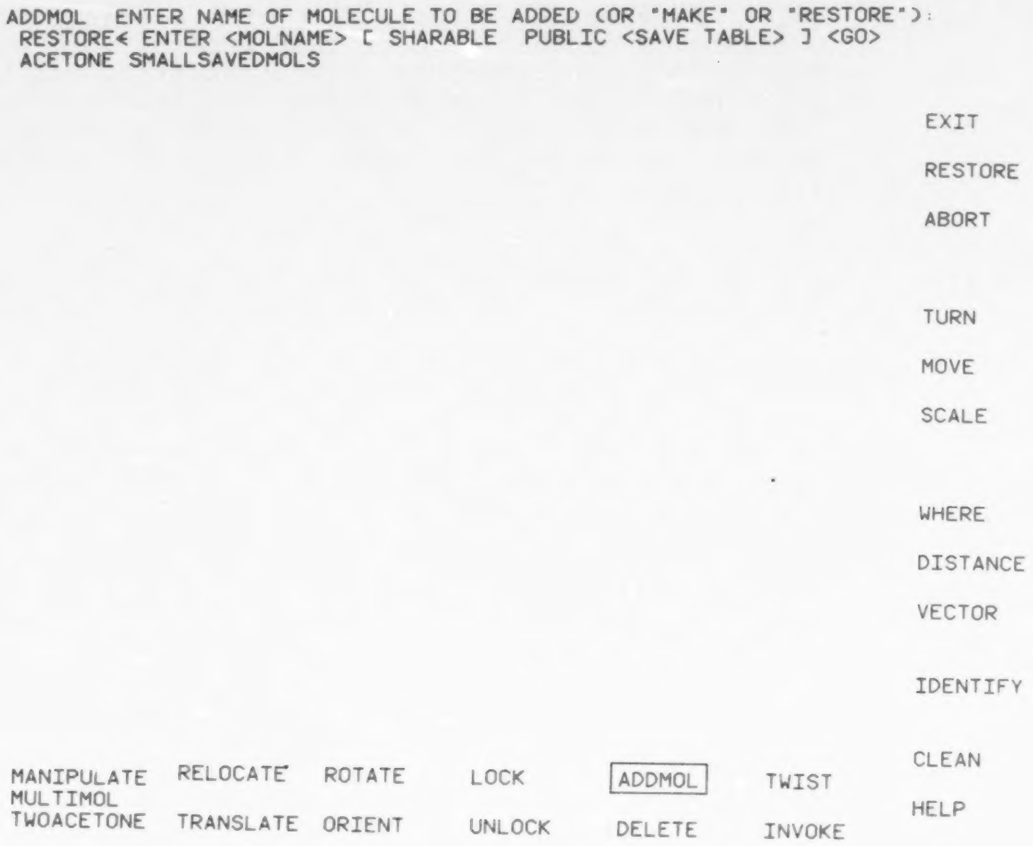
To add a molecule that is already in your workarea, type its name, followed by <GO>. Typing MAKE<GO> puts you in the MAKE MOLECULE quiz (see Part 1 of this chapter). In that case, the molecule you sketch in is added to the multimolecule after its model is computed. Typing RESTORE<GO> means that you want to restore a previously saved molecule and add it. In this case, the system then prompts:

```
ENTER <MOLNAME> [SHARABLE PUBLIC <SAVE TABLE>]<GO>
```

In response, type the name of the saved molecule, followed by where it is saved: type <GO> if it is saved in your PRIVATE SAVED table; type SHARABLE or PUBLIC followed by <GO> if the molecule is saved in SHARABLE SAVED or PUBLIC SAVED; type the name of another save table followed by <GO> if it is saved in a named save table.

MAKE MULTIMOLECULE Example

Figure 2-25 illustrates an example where our ACETONE molecule, which was saved in a named saved table called SMALLSAVEDMOLS, is to be added to an initially empty multimolecule called TWOACETONE. At this point, we have pointed to the ADDMOL light button and typed ACETONE SMALLSAVEDMOLS.



```
ADDMOL  ENTER NAME OF MOLECULE TO BE ADDED (OR "MAKE" OR "RESTORE"):
 RESTORE< ENTER <MOLNAME> [ SHARABLE  PUBLIC <SAVE TABLE> ] <GO>
 ACETONE SMALLSAVEDMOLS
```

EXIT
RESTORE
ABORT
TURN
MOVE
SCALE
WHERE
DISTANCE
VECTOR
IDENTIFY
CLEAN
HELP
MANIPULATE
MULTIMOL
TWOACETONE
RELOCATE
TRANSLATE
ROTATE
ORIENT
LOCK
UNLOCK
ADDMOL
DELETE
TWIST
INVOKE

Figure 2-25. MAKE MULTIMOLECULE: user has specified that ACETONE is to be restored from SMALLSAVEDMOLS and added to TWOACETONE.

When you have specified the molecule to be added, the screen is erased, and the new molecule is displayed THREED COMPLETE, along with a new set of light buttons, as shown in Figure 2-26. The legend

ADDING
MOLECULE
molecule-name

is displayed in the lower-left corner of the screen to remind you that you are adding a molecule. Use the TURN and ORIENT light buttons to set the molecule in a desired orientation (these commands are described below under "Manipulating One Molecule"). When the desired orientation is achieved, you can point to the INSTALL button, which actually adds the molecule to the multimolecule. The system asks you to point to an atom or bond in the molecule, and then to type in three numbers, which are the X-, Y-, and Z-coordinates in the multimolecule at which the specified atom or bond is to lie, measured in angstroms from the origin. Figure 2-27 illustrates the appearance of the screen at this point. Once you have supplied coordinates, the screen is erased, and the entire multimolecule appears, including the newly added molecule, along with all the MAKE MULTIMOLECULE quiz light buttons, as shown in Figure 2-28.

If, in the process of adding a molecule, you decide not to install it, you can point to ABORT, and the multimolecule will reappear. If you add a molecule and later decide to delete it, point to the DELETE light button, and then to an atom or bond in the molecule you wish to delete. The screen is erased and redrawn without the deleted molecule.

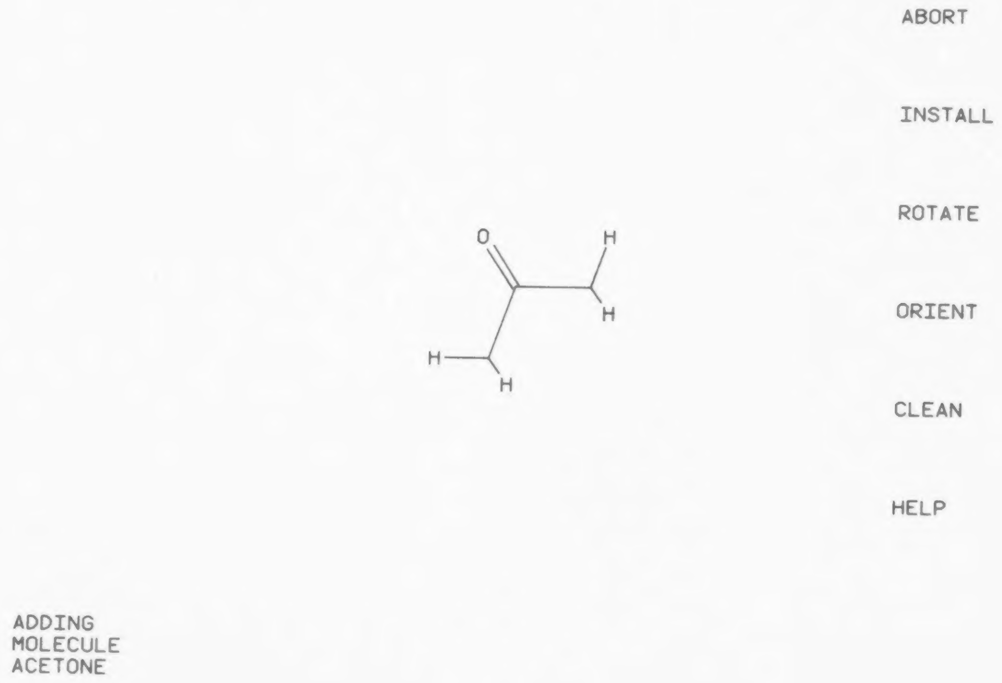
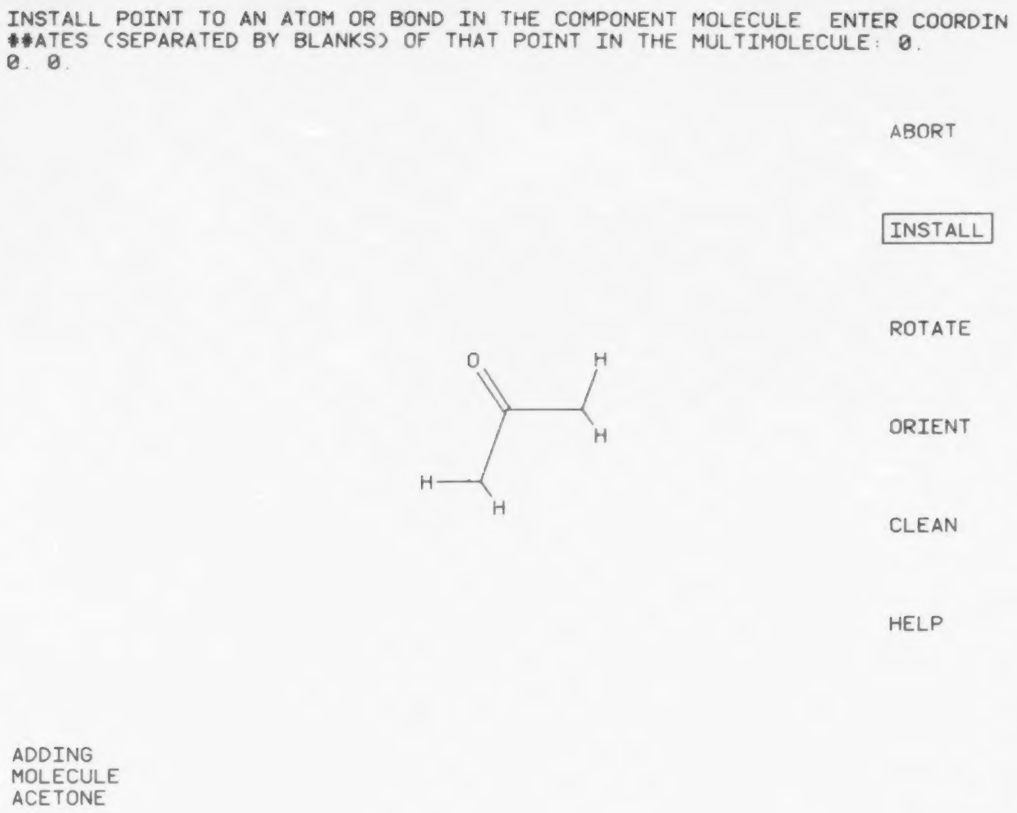


Figure 2-26. ADDING MOLECULE light buttons in multimolecule quiz.



```
INSTALL POINT TO AN ATOM OR BOND IN THE COMPONENT MOLECULE  ENTER COORDIN
**ATES (SEPARATED BY BLANKS) OF THAT POINT IN THE MULTIMOLECULE: 0.
0. 0.
```

ABORT
INSTALL
ROTATE
ORIENT
CLEAN
HELP
O
H
H
H
H

```
ADDING
MOLECULE
ACETONE
```

Figure 2-27. Display screen when using INSTALL light button.

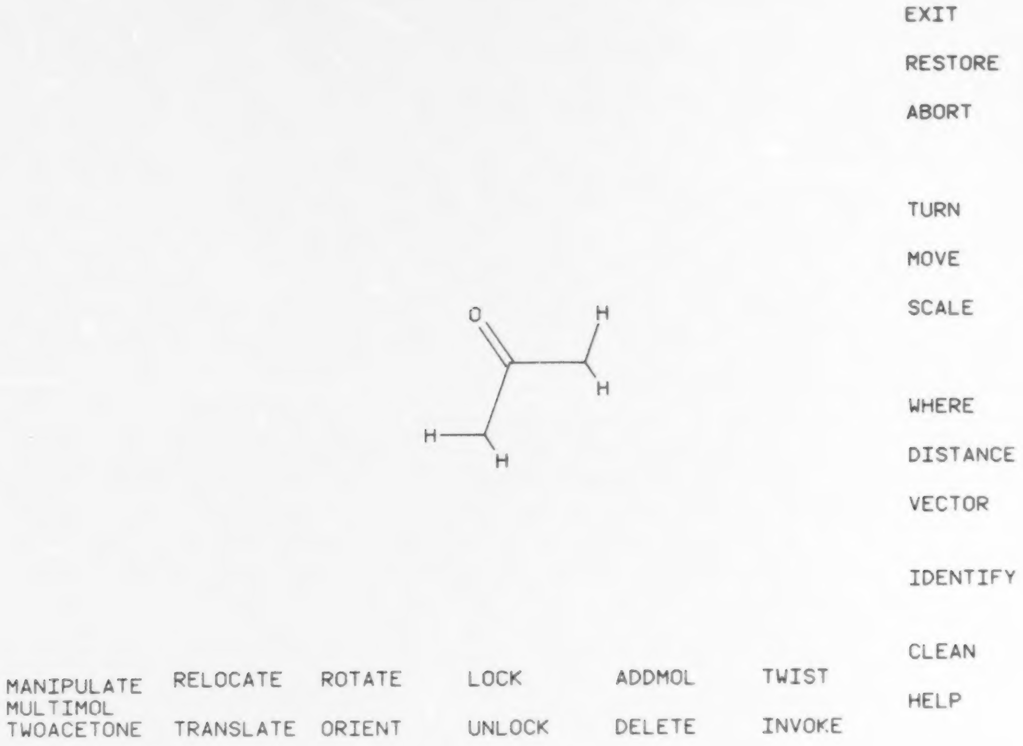

EXIT
RESTORE
ABORT
TURN
MOVE
SCALE
WHERE
DISTANCE
VECTOR
IDENTIFY
CLEAN
HELP
O
H
H
H
H
MANIPULATE
MULTIMOL
TWOACETONE
RELOCATE
TRANSLATE
ROTATE
ORIENT
LOCK
UNLOCK
ADDMOL
DELETE
TWIST
INVOKE

Figure 2-28. Multimolecule quiz example after installing ACETONE.

Manipulating the Entire Multimolecule

You can manipulate the multimolecule as a whole without changing the coordinates of any component molecules, using the SCALE, MOVE, and TURN commands. These all work just like their top-level counterparts, described in Part 3 of this chapter. The forms of the three commands are as follows:

```
SCALE [factor]

MOVE point point

           X [point]
TURN angle Y [point]
           Z [point]
           point point
```

where the meanings of "factor", "angle", and "point" are the same as the meanings described in Part 3.

Manipulating One Molecule

Several of the commands at the bottom of the display screen move one molecule with respect to the other molecules in the multimolecule reference frame. Unlike the commands above, which only alter the display appearance, these commands alter the structure of the multimolecule.

Two commands can be used to shift (translate) the position of a molecule. RELOCATE moves the molecule <u>to</u> a specified point:

```
                point2
RELOCATE point1
                x y z
```

That is, you point to an atom or bond in the molecule to be relocated ("point1") and then either point to a new location ("point2"), which results in moving the molecule parallel to the display screen, or type the 3-dimensional coordinates to move to as three numbers "x y z". In the latter case, the coordinates are expressed in the reference frame of the multimolecule. You may use the WHERE command described below to help determine suitable values for x, y and z.

A second command, TRANSLATE, is used to move a molecule by a specified amount:

```
TRANSLATE point dx dy dz
```

You point to an atom or bond in the molecule to be moved, and then type three numbers ("dx dy dz") which are the x-, y- and z-displacements, in the multimolecule's coordinate frame.

There are two commands that change the orientation of an individual molecule. The ROTATE command has the form:

```
                  X point
ROTATE angle      Y point
                  Z point
                  point point
```

The command works like TURN, except that only one molecule is affected. Thus, it is required to point to an atom or bond in that molecule of interest; that point will also be the center of rotation.

The ORIENT command operates in a similar fashion to the top-level ORIENT command described in Part 3, allowing a particular direction in the individual molecule to be aligned with a direction in the multimolecule reference frame. The command has the form:

```
           point1 point2    X
ORIENT                      Y
           point            Z
                            x y z
```

You may orient a pair of atoms in a particular molecule by pointing to them ("point1" and "point2"), or you may orient a particular bond by pointing to it ("point"). You must then specify the axis that the bond or a line between the pair of atoms you have indicated should be oriented along. If you type X, Y or Z, they are oriented along one of the three display axes. (As usual, the X-axis goes from left to right, the Y-axis from bottom to top, and the Z-axis from back to front of the screen). If you type three numbers ("x y z"), they are interpreted as a vector in the multimolecule's coordinate system, which in general is not the same as the display coordinate system, and the atom pair or bond is lined up in the direction specified by that vector. ORIENT is a very powerful command in the multimolecule quiz whose full utility can be appreciated only after considerable experimentation.

ROTATE and ORIENT also appear on the display screen when you are adding a molecule to the multimolecule. They have the same command format and effect in that context.

The last command for manipulating an individual molecule is TWIST, which allows a part of a molecule to be twisted around a single, non-ring bond; it has the form:

```
TWIST angle point1 point2
```

where you type the angle and point to the two atoms at either end of the single bond ("point1" and "point2"). TWIST rotates the part of the molecule attached to "point2" by "angle" degrees clockwise about the bond going from "point1" to "point2."

Querying the Multimolecule

The multimolecule quiz also includes several commands that type out information about a multimolecule. The command:

```
IDENTIFY point
```

types out the index in the multimolecule of the molecule containing "point" (this index generally indicates the order in which the molecules were added to the multimolecule), and, if "point" is an atom or bond, its numeric index within the individual molecule. The command

```
WHERE point
```

types the coordinates of the location pointed to. If "point" corresponds to an atom or bond, the coordinates of the atom or bond in the multimolecule reference frame are typed out. If "point" does not correspond to an atom or bond, the coordinates on the display screen of the location you pointed to (Z=0) are typed. For both the IDENTIFY and WHERE commands, you can continue to point to additional locations until you indicate another light button command.

Finally, you can ask for PROPHET to type the distance (in angstroms) between two points (atoms or bonds within the multimolecule) with the command:

```
DISTANCE point1 point2
```

Leaving the Quiz

Three commands EXIT, RESTORE, and ABORT end the multimolecule quiz. EXIT finishes the quiz normally and leaves a display of the multimolecule on the screen. RESTORE also finishes the quiz normally, but restores the display screen to its pre-quiz appearance. Finally, ABORT can be used to leave the quiz in problem situations; its effect is to restore the multimolecule to its pre-quiz state. Note that none of these commands saves a copy of the multimolecule; they leave it merely as a temporary variable in your current workarea. In order to have the multimolecule available in subsequent PROPHET sessions, you must use the SAVE command, as discussed in the context of saving molecules in Part 1 of this chapter.

Advanced Commands

PROPHET provides a number of commands for more sophisticated manipulation within the multimolecule quiz. The command:

```
VECTOR point1 point2
```

types out the vector (x, y, z) in the multimolecule reference frame from "point1" to "point2". In addition, the command "remembers" that vector for use in other commands.

The remembered vector can be referred to later by pointing to the VECTOR light button. For example, the command

```
ORIENT bond VECTOR
```

turns an individual molecule so that the specified "bond" lines up with the last previously specified vector. Similarly, the remembered vector can be used as a displacement in the command:

```
TRANSLATE point VECTOR
```

At any time, you can inspect the currently "remembered" vector by the command VECTOR <GO>.

A second set of commands can be used to impose a kind of constraint on the multimolecule, namely, to attach two or more component molecules rigidly. The effect of this attachment is that any of the individual molecule commands -- TRANSLATE, RELOCATE, ROTATE, and ORIENT -- affect not only the indicated molecule, but also all the other molecules attached to it, either directly or through a chain of rigid attachments. The command to create an attachment is:

```
LOCK point1 point2
```

This creates an attachment between the molecules containing the two points. It also has the effect of attaching any other molecules already attached to these two.

The effects of a LOCK command can be undone with the UNLOCK command:

```
        <GO>
UNLOCK  point1 <GO>
        point1 point2
```

UNLOCK <GO> removes all existing links between all the molecules in the multimolecules. Indicating UNLOCK and one point (followed by <GO>) removes all attachments to "point1." In the third form ("point1 point2") the specified attachment is removed; however, the molecules containing the points might still be indirectly attached.

Finally, INVOKE can be used to call user-written procedures which interact with the multimolecule. Since this involves molecule programming, it is described in Chapter III of this manual.

Manipulating a Multimolecule

When you wish to change the orientation of molecules in an existing multimolecule or add or delete molecules from an existing multimolecule, use the command:

```
MANIPULATE multimolecule-name [NOBACKUP]
```

This command puts you into the MAKE/MANIPULATE MULTIMOLECULE quiz, where you have available all the commands described above. Normally, when you enter the quiz with the MANIPULATE command, PROPHET makes a backup copy of your multimolecule (called something like BACKUPMULTIMOL) and saves it in your PRIVATE SAVED table. When you EXIT from the quiz, this saved multimolecule is automatically deleted, but if you terminate the quiz with ABORT, the saved backup copy is used to restore the original state of your multimolecule.

If you have a relatively large multimolecule, there may be a significant delay between the time you type the MANIPULATE command and the time the quiz actually gets started; this is due to the time it takes for the system to make and save the backup copy. If you already have a saved copy of your original multimolecule, you can save the time involved in making the backup copy by appending the word NOBACKUP to the MANIPULATE command. If you have done this, using ABORT will not restore the multimolecule copy in your workarea to its original state.

Displaying a Multimolecule

A multimolecule may be displayed by use of the command

```
DISPLAY multimolecule-name qualifiers
```

The "qualifiers" can include any of the molecule display qualifiers described in Part 3 of this chapter, excepting SELECTING and LABELING. These qualifiers are (standard defaults are underlined):

THREED	NOCOMPLETE	NODREIDING
STEREO	COMPLETE	DREIDING
NOATOMINDEX	NOBONDINDEX	TYPES
ATOMINDEX	BONDINDEX	NOTYPES

Notice that, unlike molecule displays, the default mode for multimolecule displays is THREED; in fact, SKETCH displays for multimolecules are not currently available. Multimolecule displays can be manipulated using the same commands as for molecule displays (functions are analogous). These commands are MOVE, SCALE, TURN, FREEZE, DEEPFREEZE, and TWIST. In addition, the system procedure %PERSPECTIVE, which is used for stick and ball perspective displays of molecules, may be used for multimolecules as well. Its use is described in Part 3 of this chapter.

Part 5: Disk Molecules and Multimolecules

A Method for Handling Very Large Molecules

A PROPHET molecule consists, in part, of two PROPHET tables which describe the molecule's atoms and bonds and optionally a third table which describes the molecule's constraints if any are present (see Chapter III of this manual for details). Normally, all such tables in molecules are core-based *temporary* tables: they are stored in the user's workspace, and are not preserved at LOGOUT. Since these tables reside in the work area, building a large molecule or multimolecule can result in the error message "125. SHORT OF SPACE..."

The atoms, bonds, and constraints tables of *disk molecules*, on the other hand, are permanent (disk-based) tables. Because these tables (which form most of a molecule's bulk) are stored on disk, disk molecules (and disk multimolecules, composed of disk molecules), can grow pretty much without bounds. Of course, one doesn't get such a capability without penalty, and manipulations involving disk molecules and multimolecules tend to be somewhat slower than for their in-core counterparts (known as *core molecules* and *core multimolecules*). Although you must be the ultimate judge, a reasonable guideline is that a molecule or multimolecule should be disk-based only if it contains more than about 400 atoms, or if it contains more than about 150 constraints.

In most cases, disk molecules/multimolecules behave exactly like core molecules. Where differences occur, they generally involve situations in which the molecules/multimolecules are to be saved or restored from other tables.

Making a Disk Molecule

Disk molecules can be created using the same interactive MAKE MOLECULE quiz used to make core molecules. The full form of the MAKE MOLECULE command is:

```
        [CORE]
MAKE            MOLECULE unused-name [FROM molecule-name
        [DISK]

[SELECTING...][NOQUIZ]]
```

The important difference between this and the simple MAKE MOLECULE command is the addition of the optional qualifiers CORE and DISK. If CORE is specified, then "unused-name" will be a core molecule. If DISK is specified, "unused-name" will be a disk molecule. If neither qualifier is specified, "unused-name" will be a core molecule, unless FROM molecule-name specifies an existing disk molecule. In all other respects, the MAKE MOLECULE quiz behaves in the same manner for disk molecules as it does for core molecules as described in Part 1 of this chapter.

Binding and Unbinding Disk Molecules

A major complication in handling disk molecules is the fact that they cannot be stored in other tables (such as your PRIVATE SAVED variables table) since they contain disk-based tables. For this reason, we introduce the concept of *binding*: When you BIND a molecule, its disk-based tables become directly accessible for all the usual molecule manipulations. However, to place the disk molecule in another table, one must first *UNBIND* it, replacing its constituent tables with the names under which those disk-based tables could be subsequently accessed. The SAVE command described below automatically takes care of unbinding the disk molecule for you. However, if you intend to issue the command:

```
SET ROW row-id COL col-id OF table-name TO disk-mol-name
```

you must first unbind "disk-mol-name" by issuing the command:

```
       PRIVATE
UNBIND SHARABLE disk-mol-name
       PUBLIC
```

Notice that PUBLIC or SHARABLE must be specified if you do not wish the tables to be PRIVATE. The type specifies which directory the tables are placed in, and should match the type of the table into which the disk molecule itself is being SET. PROPHET advises you if you are UNBINDing a molecule that already has permanent tables from a previous UNBIND, and gives you the option of deleting the old tables.

Naturally, there is an inverse operation to unbinding, namely <u>binding</u>: When the BIND command is applied to a disk molecule, the molecule is reconnected to its permanent tables. This can be done with the command:

```
BIND disk-mol-name
```

Notice that it is not necessary to specify whether the tables for the disk-molecule are PUBLIC, PRIVATE or SHARABLE in the BIND command. BIND types out some information about the permanent tables and gives you the option of dearchiving any of those tables, if they are currently archived (the molecule cannot be used until this is done). BIND issues an error message if any of the tables for the disk molecule cannot be found in the system.

Note that the RESTORE command described below automatically issues a BIND command when a disk molecule is restored from a saved variables table. You must use the BIND command explicitly, however, when you wish to use a molecule that has been extracted from a table with a command like:

```
disk-mol-name = ROW I COL J OF table-name
```

Saving and Restoring Disk Molecules

Disk molecules can be saved and restored from save tables using the usual SAVE and RESTORE commands (see Part 1 of this chapter). There are a few extra messages and questions that may be typed when saving/restoring disk molecules: When you SAVE a disk molecule that has been previously saved, PROPHET advises you of what tables had been associated with the previously saved version, and whether those tables had been protected and/or archived. The SAVE command then asks whether you wish to delete those tables, since they will no longer be associated with this molecule. PROPHET then makes new permanent tables for the molecule and reports the names of the tables to you.

When you RESTORE a disk molecule, PROPHET again advises you of the names and status of the permanent tables associated with the molecule, and, if any are archived, asks whether you wish them to be dearchived (which they must be for the molecule to be used). In addition, if any of the tables associated with the molecule have been removed from the system (not found in the appropriate directory or archive directory), you are advised of this with an error message; in that case, you should consult your PROPHET representative for further help. Do not be discouraged if RESTORE is somewhat slow, since it involves making copies of the molecule tables.

Deleting Disk Molecules

To remove any molecule from your workspace without affecting any saved copy, use the DELETE command. You can also remove a disk molecule from a saved table with the DELETE SAVED command (see the _PROPHET User's Manual_). In both cases, however, the permanent tables are _not_ deleted, and PROPHET informs you of this. To delete a disk molecule _and_ its associated permanent tables, use the command:

```
SUPERDELETE disk-mol-name
```

for a disk molecule in the current workspace, or:

```
            PRIVATE
SUPERDELETE SHARABLE SAVED disk-mol-name
            PUBLIC
```

for a saved disk molecule.

Protecting Disk Molecules

Finally, you might wish to protect the permanent tables of a disk molecule against inadvertent change. You could use the PROTECT command on each of the tables individually (see the _PROPHET User's Manual_), but it is safer and more convenient to say:

```
                                SELF
PROTECT disk-mol-name AGAINST PROJECT
                                SYSTEM
```

This protects any non-archived permanent tables of the molecule as specified.

Disk Multimolecules

Disk multimolecules differ from ordinary multimolecules only in that one or more of their constituent molecules is a disk molecule, rather than a core molecule. Variations of the disk molecule manipulating commands described above can be applied to disk multimolecules; in essence, their effect is to perform the specified operation on each of the constituent molecules of the multimolecule. Each of the commands already covered (SAVE/RESTORE, DELETE/DELETE SAVED, SUPERDELETE/SUPERDELETE SAVED, PROTECT, and BIND/UNBIND) can thus also be applied to disk multimolecules.

To invoke the MAKE MULTIMOLECULE quiz, one uses the command whose full form is:

```
             [CORE]
     MAKE              MULTIMOLECULE unused-name
             [DISK]
```

If CORE or DISK is specified, any molecules added during the course of the quiz are converted to the specified type, and the resulting multimolecule has that type as well. If neither qualifier is included, the resulting multimolecule is a disk multimolecule if any molecules added during the quiz are disk molecules.

best copy available

Part 6: Representation of Empirical Chemical Formulas

Empirical Formula Standards

In PROPHET empirical chemical formulas (both those associated with every molecule and any that may have been independently created) are stored in a data type called CHEMTEXT. CHEMTEXT formulas include only formulas that are consistent with the notation developed 75 years ago by the U.S. Patent Office and presently used by such authorities as the Chemical Abstracts Service. Each CHEMTEXT formula must fulfill each of the following four conditions:

1. Elements are listed in the following order:
    a. For Carbon-containing compounds, C is first, H second (if present) and everything else is in alphabetical order.
    b. For compounds not containing Carbon, all elements (including H if present) are in alphabetical order.
    c. Element symbols ? and E (see point 3.) are treated as special cases; they are placed at the end of the list of elements. If both appear in the same molecule, ? is placed before E at the end of the formula.
2. If an element appears more than once in a molecule, its frequency is indicated by a number inserted immediately after the element. Frequencies from 1 to 99999 are recognized, but if an element is present exactly once, the frequency is typically not explicitly shown (the digit one, however, is tolerated if present in such a case).
3. Only element symbols (one or two characters) found in the PROPHET elements table (see Appendix A of this manual) are recognized. All standard element symbols are maintained in the PROPHET element table, but certain other symbols are also allowed; in particular, use of symbols '?', 'E' and 'N+' is supported (see Chapter II, Part 1).
4. To make potentially confusing character strings (such as 'CLINPU') unambiguous, one blank is inserted after each element unit (including the frequency number, if any). The last blank after the last unit may be omitted, however.

These conventions allow us to provide a capability for ordering CHEMTEXT formulas according to the conventions utilized in *Chemical Abstracts* for sorting empirical chemical formulas. Sorting is done alphabetically by element symbols and numerically by element frequencies, each being treated as a unit from left to right. An example of the ordering that results is:

$B\ H_4$

$C\ O_2$

$C_6\ H_5\ O$

$C_6\ H_6$

$C_{10}\ H_6\ O_3$

$C_{10}\ N_3\ O_5$

$CL_2\ CO$

$H_2\ O$

Making CHEMTEXT Formulas

Every PROPHET molecule has a CHEMTEXT formula associated with it that has automatically been computed by the system. Independent CHEMTEXT formulas may also be made by using the following command:

```
MAKE CHEMTEXT formula-name <GO>
```

After you type this command, you are asked to type a CHEMTEXT formula. If it is not a proper formula according to the rules explained above, MAKE CHEMTEXT interactively helps you to correct it until it has become a proper formula. Likewise, if you are using the SET or FILL command to enter values into table columns that are of type CHEMTEXT, you are interactively aided until you succeed in entering formulas that can be made to obey the rules for proper CHEMTEXT formulas.

Sorting CHEMTEXT Formulas

In order to sort a table by a column containing CHEMTEXT formulas, you can use the usual SORT command as follows:

```
SORT table-name BY COL col-id
```

where COL col-id contains CHEMTEXT formulas.

The CHEMTEXT ordering conventions are utilized at all levels. If, for example, you have a table called FORMS in which column 1 is of type CHEMTEXT, the command

```
MAKE TABLE FORMS2 FROM FORMS FOR COL 1 > 'C2'
```

would create a table FORMS2 containing only those rows for which the formula in column 1 would come after the formula 'C2' in the sort sequence. For example, rows containing formulas such as 'C2 H6 O', 'C18 H30 ' and 'H2 O' in column 1 would be included whereas rows containing formulas such as 'B H4', 'C O2' and 'C2' would not. Similarly the command

```
MAKE TABLE FORMS2 FROM FORMS FOR COL 1 CTNS 'C2'
```

would include only those rows with formulas in column 1 containing exactly two carbon atoms. Note that the expression "CTNS 'C'" means that you want formulas that contain exactly one carbon and not formulas that contain more than one carbon atom.

3. Functions and Procedures

3. Functions and Procedures

Chapter III.

MOLECULE FUNCTIONS AND PROCEDURES

The PL/PROPHET programming language described in the PL/PROPHET User's Guide facilitates sophisticated programming involving chemical information by providing molecules and multimolecules as data types. The PROPHET system provides a large number of built-in tools for manipulating molecules and multimolecules, as described in Chapter II of this manual. This chapter describes how the user can extend the repertoire of molecule tools by writing procedures that create, modify, access, and display molecular information.

Part 1 describes the MOLECULE data type, which consists mainly of three specially structured TABLEs. Part 2 describes the molecule gatekeepers, a set of procedures which facilitate interaction with the molecule data type in an intuitive, controlled, yet efficient way. Part 3 describes other molecule procedures. Part 4 describes the structure and manipulation of multimolecules, particularly the INVOKE command which can be used to extend the capabilities of the MAKE/MANIPULATE MULTIMOLECULE commands. Part 5 presents some examples of programs involving molecules and multimolecules.

Table 3-1: Components of Molecule Structure

Subfield Name	Type	Contents
ATOMS	TABLE	One atom description per row, specifying type, attached bonds (by index in the BONDS table) and coordinates.
BONDS	TABLE	One bond description per row, specifying type, and atoms attached to either end of bond (by index in the ATOMS table).
CONSTRAINTS	TABLE	One constraint description per row, specifying type, atoms and bonds involved, and value.
ATTRIBUTES	ANY ARRAY ARRAY	List of user-supplied attributes, accessed by name of attribute.
FLAGS	BIT	Set of bit switches, describing various properties of molecule.
FORMULA	CHEM-TEXT	The empirical chemical formula for the molecule.
NAMES	TEXT ARRAY	Names for the molecule, supplied by the user during MAKE MOLECULE.
NOTES	TEXT	User-supplied text.
%TRANS	FLOAT ARRAY ARRAY	Transformer applied to display of molecule's 3D model.

Part 1: The Molecule Data Type

PROPHET molecules have two features which make them simple to use in PL/PROPHET programs:

1. Molecules are TABLE-based; almost all the information in the MOLECULE is stored in a set of three tables which are subfields of the MOLECULE data type. This has the advantage of presenting you with a structure with considerable flexibility and robustness that you are already familiar with.

2. Access to molecules is handled via gatekeeper procedures. Gatekeepers are safe, convenient tools that relieve you from worrying about many of the details of the molecule's computer-based representation. They protect you against making inconsistent changes in the molecule's data structure, and also carry out the modifications you wish to make more efficiently than you could if you wrote your own procedures to modify parts of the data structure directly. For example, to add a single bond between atoms A1 and A2 of molecule M, you would write

   ```
   CALL %ADDBOND(M,'S',A1,A2,IBND)
   ```

   (where IBND would be set by %ADDBOND to the FIXED index of the new bond).

Table 3-1 summarizes the components of a PROPHET molecule. The subfields of the MOLECULE data type can be referred to in the same way as the subfields of any complex data type, as described in the PL/PROPHET User's Guide. For example, the ATOMS subfield of the molecule ACETONE can be referred to either as

```
ATOMS OF ACETONE
```

or

```
ACETONE.ATOMS
```

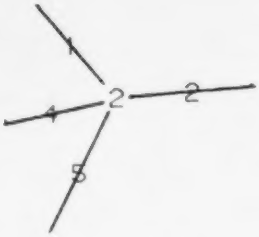
The remainder of this part describes the subfields of a MOLECULE in some detail.

The ATOMS Table

The ATOMS subfield of a molecule is a TABLE in which each row describes a single atom (Table 3-2 describes its structure). The <u>index</u> of an atom is simply the FIXED number of its row in the ATOMS table; atoms are usually referred to by their indices. Column 1 of the ATOMS table is the <u>type</u> of the atom, that is, a TEXT indicating the element type, such as 'H', 'C', or 'O'. Column 2 describes the connectivity of the molecule by listing for each atom the indices of all attached bonds in a FIXED ARRAY, e.g.,

ATOMS 8R X 9C

	1 TYPE	2 BOND ARRAY
2	C	[1,2,4,5]



Columns 3 and 4 are the <u>sketch</u> <u>coordinates</u> of the atom -- its position in the two-dimensional (2D) sketches produced by MAKE MOLECULE, expressed in arbitrary units.

Columns 5 to 7 are the <u>model</u> <u>coordinates</u> of the atom, representing its position in the three-dimensional (3D) models created by the COMPUTE MODEL command or supplied by you; "Model Space" is described in units of Angstroms.

While columns 1 through 7 of the ATOMS table are reserved by PROPHET for the purposes described above, you may add other columns for other "per-atom" data; for example, column 0 (ROWNAMES) can be used to store symbolic atom names. Other columns can be used for selectively displaying and making submolecules (see Part 3 of this chapter for uses of the ATOMS table columns in molecule displays). Note that the Wipke energy minimization program adds a column to the ATOMS table with the heading 'WIPKE'; hence, you should not use that heading for any column you add yourself.

To examine the contents of the ATOMS table, use the command:

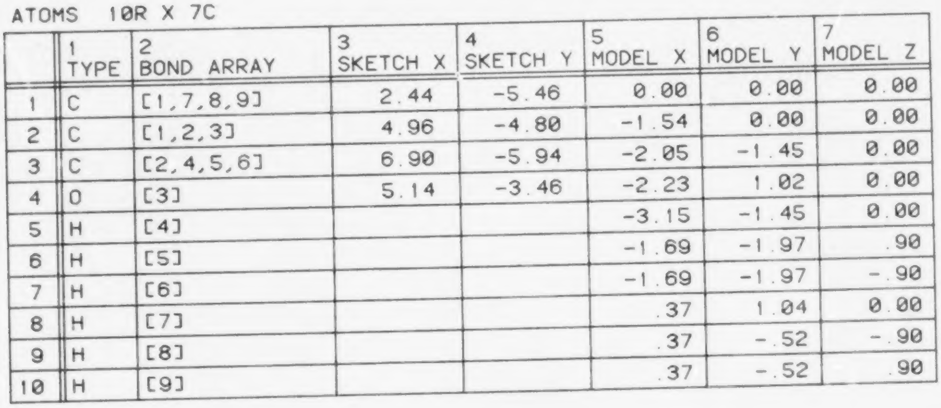
```
DISPLAY ATOMS OF molecule-name
```

The ATOMS table for the molecule ACETONE made in Part 1 of Chapter I of this manual is reproduced as Figure 3-1.

Table 3-2: Structure of the ATOMS Table

Col. No.	Col. Name	Col. Type	Contents of Column
1	TYPE	TEXT	Element type of atom (e.g, 'C', 'H', etc.).
2	BOND ARRAY	FIXED ARRAY	Indices of bonds connected to atom.
3	SKETCH X	FLOAT	x-coordinate of atom in 2D sketch.
4	SKETCH Y	FLOAT	y-coordinate of atom in 2D sketch.
5	MODEL X	FLOAT	x-coordinate of atom in 3D model.
6	MODEL Y	FLOAT	y-coordinate of atom in 3D model.
7	MODEL Z	FLOAT	z-coordinate of atom in 3D model.

Model coordinates are expressed in the local coordinate frame of the molecule (units=angstroms).



ATOMS 10R X 7C

	1 TYPE	2 BOND ARRAY	3 SKETCH X	4 SKETCH Y	5 MODEL X	6 MODEL Y	7 MODEL Z
1	C	[1,7,8,9]	2.44	-5.46	0.00	0.00	0.00
2	C	[1,2,3]	4.96	-4.80	-1.54	0.00	0.00
3	C	[2,4,5,6]	6.90	-5.94	-2.05	-1.45	0.00
4	O	[3]	5.14	-3.46	-2.23	1.02	0.00
5	H	[4]			-3.15	-1.45	0.00
6	H	[5]			-1.69	-1.97	.90
7	H	[6]			-1.69	-1.97	-.90
8	H	[7]			.37	1.04	0.00
9	H	[8]			.37	-.52	-.90
10	H	[9]			.37	-.52	.90

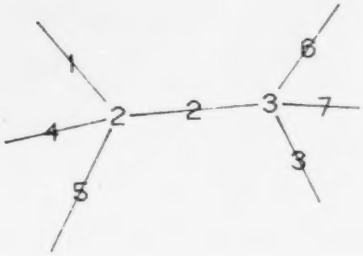
Figure 3-1. ATOMS table of ACETONE molecule.

The BONDS TABLE

The BONDS subfield of a molecule is a TABLE in which each row describes one bond of the molecule (Table 3-3 describes its structure). The *index* of a bond is the FIXED number of its row in the BONDS table; like atoms, bonds are usually referred to by their indices. Column 1 of the BONDS table is the text *type* of the bond, 'S' (single bond), 'D' (double bond), 'T' (triple bond), or 'R' (resonant bond). Columns 2 and 3 describe the connectivity of the molecule by listing the FIXED indices of the two atoms at either end of the bond, e.g.,

: BONDS 7R X 5C

	1 TYPE	2 ATOM 1	3 ATOM 2
2	S	2	3



Columns 1 through 3 of the BONDS table are reserved by PROPHET for the above purposes; other columns may be added to store "per-bond" data, as described for the ATOMS table above (see Part 3 below for uses of the BONDS table columns in molecule displays). Once again, the column heading 'WIPKE' is reserved for the use of the Wipke energy minimization program.

To examine the contents of the BONDS table, use the command:

```
DISPLAY BONDS OF molecule-name
```

The BONDS table for our sample ACETONE molecule is illustrated in Figure 3-2.

Table 3-3: Structure of the BONDS Table

Col. No.	Col. Name	Col. Type	Contents of Column
1	TYPE	TEXT	Type of bond ('S', 'D', 'T', 'R').
2	ATOM 1	FIXED	Index of atom connected to one end of bond.
3	ATOM 2	FIXED	Index of atom connected to other end of bond.

BONDS 9R X 3C

	1 TYPE	2 ATOM 1	3 ATOM 2
1	S	1	2
2	S	2	3
3	D	2	4
4	S	3	5
5	S	3	6
6	S	3	7
7	S	1	8
8	S	1	9
9	S	1	10

Figure 3-2. BONDS table of ACETONE molecule.

The CONSTRAINTS Table

The CONSTRAINTS subfield of the molecule is a TABLE which is used to store *constraint* information which guides the PROPHET model builder in computing 3D models (Table 3-4 describes its structure). Each row of the CONSTRAINTS table describes a single constraint, which is referred to by its *index*, the FIXED number of that row. Column 1 of the constraints table is the *type* of the constraint. The PROPHET model builder recognizes the constraint types 'BONDLENGTH', 'TORSION', 'BOND ANGLES', 'DIRECTIONS', and 'CHIRALITY'; you can set other types, but they are ignored by the model builder. Columns 2 and 3 list the atoms and bonds involved in the constraint, as FIXED ARRAYs of atom/bond indices. Column 4 contains the *value* of the constraint; its interpretation depends on the type of the constraint, as shown in Table 3-5.

To examine the CONSTRAINTS table, use the command:

```
DISPLAY CONSTRAINTS OF molecule-name
```

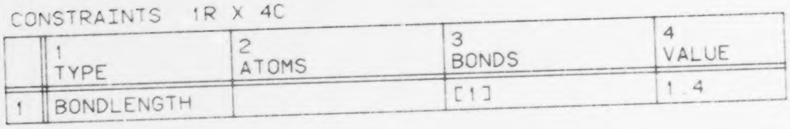
If we had typed the command:

```
CONSTRAIN LENGTH OF BOND 1 OF ACETONE TO 1.4
```

and then displayed the CONSTRAINTS table of ACETONE, the resulting table would be that illustrated in Figure 3-3.

Table 3-4: Structure of the CONSTRAINTS Table

Col. No.	Col. Name	Col. Type	Contents of Column
1	TYPE	TEXT	Type of constraint (e.g., 'BONDLENGTH', etc).
2	ATOMS	FIXED ARRAY	Indices of atoms involved in this constraint.
3	BONDS	FIXED ARRAY	Indices of bonds involved in this constraint.
4	VALUE	ANY	Value to which specified atoms and bonds are constrained.



CONSTRAINTS 1R X 4C

	1 TYPE	2 ATOMS	3 BONDS	4 VALUE
1	BONDLENGTH		[1]	1.4

Figure 3-3. CONSTRAINTS table of ACETONE molecule after typing command: CONSTRAIN LENGTH OF BOND 1 OF ACETONE TO 1.4

Table 3-5: System-Recognized CONSTRAINT Types

Constraint Type	Atoms Col Usage	Bonds Col Usage	Value Col Usage
BONDLENGTH	not used	Bond whose length is constrained.	Length in angstroms.
TORSION	not used	Three adjacent bond defining torsion angle.	Torsion angle (dihedral angle) in degrees.
CHIRALITY	atom at chiral center	Three bonds adjacent to center	'CC' or 'CW', sense of circulation of bonds about center atom.
BOND ANGLES	common atom	Two, three or four bonds to a common atom whose angles are constrained.	One, three or six bond angles angles in degrees.
DIRECTIONS	common atom	Two or more bonds to a common atom whose directions are constrained.	Two or more vectors defining relative direction of each bond.

The ATTRIBUTES Subfield

The ATTRIBUTES subfield of the molecule is an ANY ARRAY ARRAY which you may use to store arbitrary information about the molecule. This information is accessed by attribute name, rather than index (as for atoms, bonds, and constraints), allowing greater convenience and flexibility. The first "column" of the ATTRIBUTES array contains the names of the attributes, while the second "column" contains the values associated with the attributes. For example, if molecule M has the value '6.3' associated with the attribute 'ACTIVITY', then, for some FIXED value I, M.ATTRIBUTES(I,1) = 'ACTIVITY' and M.ATTRIBUTES(I,2) = 6.3.

To examine the attributes of a molecule, use the commands:

```
EAD
DISPLAY
PRINT    ATTRIBUTES OF molecule-name
TYPE
```

For example, if you retrieve the molecule AN01 from the public table ANDR1 as described in Appendix B of this manual, the command

```
TYPE ATTRIBUTES OF AN01
```

produces the following output:

```
CELLCONSTANTS
11.528
11.128
7.645
90.
110.61
90.0

SPACEGROUP
P21

REFERENCE
N. MANDEL ET. AL., ACTA CRYST. B28, 308 (1972)
```

The FLAGS Subfield

The FLAGS subfield of the molecule contains a number of binary switches describing the molecule. The switches have the following symbolic names:

FLAG	MEANING
%M%FRAG	TRUE for fragment molecule
%M%MDL	TRUE if any atoms have 3D model coordinates
%M%SKT	TRUE if any atoms have 2D sketch coordinates
%M%DSK	TRUE for disk molecules
%M%UNBD	TRUE for disk molecules in the UNBIND state

The flags are set by various system procedures, but can be accessed and modified by the gatekeepers %GETFLAG and %SETFLAG described in Part 2 of this chapter.

The FORMULA, NAMES, NOTES and %TRANS Subfields

The FORMULA subfield is a CHEMTEXT which represents the molecule's empirical formula. It is automatically computed by the system. The NAMES subfield of the molecule is a TEXT ARRAY which is a list of names you have supplied for the molecule. This array is set by the epilogue of the MAKE MOLECULE quiz, and may be altered at will by the user. The NOTES subfield of the molecule is a TEXT which you may use to store any information about the molecule. The %TRANS subfield is a transformer (FLOAT ARRAY ARRAY) that is applied to the molecule when its model is displayed with the THREED or STEREO qualifier. It may be manipulated by the matrix procedures and operators described in Chapter IV of this manual. These subfields may be examined by typing commands of the form:

```
        FORMULA
DISPLAY NAMES    OF molecule-name
        NOTES
        %TRANS
```

If we take the ACETONE molecule we made in Part 1 of Chapter II as an example once again, the command:

```
DISPLAY FORMULA OF ACETONE<GO>
```

produces the display:

$C_3 H_6 O$

The command:

```
DISPLAY NAMES OF ACETONE
```

produces the display:

```
    1R X 1C
  | 1
--+--------
1 | ACETONE
```

Finally, the command:

```
DISPLAY %TRANS OF ACETONE
```

produces the display:

```
    4R X 3C
  | 1   2   3
--+-----------
1 | 1.  0.  0.
2 | 0.  1.  0.
3 | 0.  0.  1.
4 | 0.  0.  0.
```

We did not use the NOTES subfield for anything in our ACETONE molecule, but we could with a command like the following:

```
SET NOTES OF ACETONE TO 'THIS IS A SAMPLE NOTE'
```

Part 2: Mole

The Concept of Gatekeepers

This section de
gatekeepers, i.e.,
access and modify the
it is possible in p
by TABLE-manipulation
STRONGEST POSSIBLE
molecules via the
preserves the inte
they perform appro
addition, they provi
of access, which is
efficient than use
all their own consis

Section 2A

This section de
used to access in
MOLECULE; these are
the variable names
of ATOMS table gatek
example of these
present them here:

Variable Name	Data T
M	MOLECU
I	FIXED
N	TEXT
T	TEXT
V	VECTO (FLOA

cule Gatekeeper Procedures

scribes in detail the molecule
those procedures which are used to
molecule data structure. While
inciple to alter the data directly
commands, it is advised IN THE
TERMS that all programs access
gatekeepers. Using gatekeepers
rity of the data structure because
riate consistency checks. In
e a convenient and mnemonic method
n fact likely to be much more
-written procedures that try to do
ency checking.

ATOMS Table Gatekeepers

cribes the gatekeepers which are
ormation in the ATOMS table of a
summarized in Table 3-6. Some of
used in the following descriptions
epers are used in more than one
atekeeper calls. For brevity, we

ype

LE

	Atom index
	Atom name
	Atom type
ARRAY)	Sketch or model coordinates

Table 3-6: ATOMS Table Gatekeeper Procedures

Calling Form	Usage
CALL %ADDATOM(M,T,I)	Adds atom of type T to molecule M; sets I to index of new atom.
%ADDNEWATOM(M,T,N,V)	Adds atom of type T with name N, sketch and model coordinates V to molecule M; returns (FIXED) value of atom index.
CALL %DELETEATOM(M,I)	Deletes atom I of molecule M; adjusts all atom index references in molecule.
CALL %DELETEATOMS(M,D)	Deletes atoms whose indices are in D from molecule M.
%LASTATOM(M)	Returns index of last atom of M.
CALL %SETUPATOMNAMES(M)	Enables atoms in molecule M to be given names.
%GETATOMNAME(M,I)	Returns (TEXT) name of atom I of M.
CALL %SETATOMNAME(M,I,N)	Sets the name of atom I of M to N.
%ATOMNAME2INDEX(M,N)	Returns index of the atom named N in molecule M.
%ATOMNAMES2INDICES(M,NSPEC)	Returns the (FIXED ARRAY) indices of those atoms specified by NSPEC in M.
%GETATOMTYPE(M,I)	Returns (TEXT) type of atom I in M.
CALL %SETATOMTYPE(M,I,T)	Sets type of atom I of molecule M to the TEXT T ('C', 'H', etc.).
%GETBONDA(M,I)	Returns (FIXED ARRAY) indices of bonds connected to atom I of molecule M.
%GETSVEC(M,I)	Returns (VECTOR) sketch coordinates of atom I in molecule M: [X Y 0]

Table 3-6: ATOMS Table Gatekeeper Procedures (cont'd)

Calling Form	Usage
CALL %SETSVEC(M,I,V)	Sets 2D sketch coordinates of atom I of molecule M to V(1), V(2).
%GETMVEC(M,I)	Returns (VECTOR) model coordinates of atom I in molecule M: [X Y Z]
CALL %SETMVEC(M,I,V)	Sets 3D model coordinates of atom I of mol M to V(1), V(2), V(3).

Notation Conventions:

M	MOLECULE	To access or modify
I	FIXED	Atom Index
N	TEXT	Atom name
D	FIXED ARRAY	Indices of atoms to be deleted
NSPEC	TEXT	Atom name specification (see text)
T	TEXT	Atom type
V	VECTOR	Model or sketch coordinates

Adding and Deleting Atoms

To add an atom of type T (where T is a TEXT such as 'H' or 'C') to molecule M, use the procedure:

```
CALL %ADDATOM(M,T,I)
```

After adding the atom, %ADDATOM sets the FIXED variable I to the index of the new atom. This procedure does not set any bonding information, nor does it fill in any sketch or model coordinates in the ATOMS table.

If you wish to set sketch and model coordinates when you add a new atom or give the new atom an arbitrary atom name (see the paragraph below on atom names), use the procedure %ADDNEWATOM. This function, which can be invoked by a statement like

```
I = %ADDNEWATOM(M,T,N,V)
```

returns a FIXED value which is the index of the new atom. You must specify the atom type in the TEXT T, and you can also specify an atom name in the TEXT N. V is a 3-VECTOR (see Chapter IV for a discussion of VECTORS) that you can use to specify the model coordinates ([X, Y, Z]) of the new atom. %ADDNEWATOM uses the first two elements of V as the new atom's sketch coordinates.

You can delete the atom with index I from M with the procedure:

```
CALL %DELETEATOM(M,I)
```

If you wish to delete more than one atom from a molecule, it is much more efficient to use the procedure %DELETEATOMS than it is to make multiple calls to %DELETEATOM. Type:

```
CALL %DELETEATOMS(M,D)
```

where D is a FIXED ARRAY containing the indices of the atoms to be deleted.

When a molecule has been altered by %ADDATOM, %DELETEATOM, %ADDNEWATOM or %DELETEATOMS, the routine %ADJUSTMOL (see Part 3 below) must be called to update certain internal subfields of the molecule. Failure to do so may render the molecule undisplayable.

WARNING!

DO NOT attempt to delete an atom by simply deleting the corresponding row of the ATOMS table. Since all indices greater than the one deleted are decreased by one, you should use %DELETEATOM, which maintains consistency by changing all data in the molecule which refer to atom indices.

Finding and Changing Atom Information

Several gatekeepers set or retrieve the information stored in the ATOMS table of a molecule. The function

```
I = %LASTATOM(M)
```

returns a FIXED number which is the index of the last atom in the molecule M. All ATOMS table gatekeepers check the validity of their atom index argument, and abort if it is less than one or greater than %LASTATOM(M).

Atom Names

The ROWNAMES column of the ATOMS table can be used to assign arbitrary names to each atom in a molecule. You may, for example, wish to name the alpha carbon of the first amino acid residue in a polypeptide 'C1A'. If you give each atom in a molecule a unique name, you can refer to atoms by atom name instead of by atom index in a number of PROPHET system procedures and public procedures.

Before you can assign names to the atoms in a molecule, you must call the procedure

```
CALL %SETUPATOMNAMES(M)
```

This sets the heading of the ROWNAME column (column 0) of the ATOMS table to ATOM NAMES and enables the values the other atom name gatekeepers set in that column to be displayed.

The function

```
N = %GETATOMNAME(M,I)
```

sets N to the name of the atom with index I in molecule M. If atom I does not have a name, the value returned is EMPTY.

To set the name of atom I of M to the TEXT value N, use the procedure

```
CALL %SETATOMNAME(M,I,N)
```

The function

```
I = %ATOMNAME2INDEX(M,N)
```

returns the FIXED index of the atom in molecule M that has the name contained in N. If no atom has that name, the value 0 is returned.

Similarly, the function

```
IARRAY = %ATOMNAMES2INDICES(M,NSPEC)
```

returns a FIXED ARRAY of the indices of the atoms in M specified in the TEXT NSPEC. NSPEC can be used to specify which atoms are desired in several ways:

1. NSPEC may contain the text string 'ALL', in which case an array of all the atom indices in the molecule is returned.
2. NSPEC may contain a phrase of the sort 'atom-name-1 TO atom-name-2' to indicate a range of atoms whose indices are returned.
3. NSPEC may also contain a list of atom names whose indices are to be returned, separated by commas.
4. Finally, NSPEC may contain arbitrary combinations of the forms mentioned in points 2 and 3, as in the sample TEXT 'C12, C13, C21 TO C33, 40, 45'. As this example illustrates, it is also possible to refer to an atom by the TEXT equivalent of its index rather than its name in NSPEC.

Atom Types

The function

```
T = %GETATOMTYPE(M,I)
```

returns a TEXT which is the type of the Ith atom of molecule M (such as 'C', 'H', 'O', or even '?' for a fragment attachment point). The type of the Ith atom of M can be set to the TEXT T by the procedure

```
CALL %SETATOMTYPE(M,I,T);
```

Connected Bonds

The function

```
BARRAY = %GETBONDA(M,I)
```

returns a FIXED ARRAY which contains the indices of all bonds connected to the Ith atom of molecule M. There is no complementary %SETBONDA, since the gatekeeper procedure %ADDBOND (see Section 2B below) alters the BOND ARRAYs of the atoms connected to any added bond in a consistent fashion.

Sketch Coordinates

The sketch coordinates of an atom are its position in the 2D sketch produced by MAKE MOLECULE. A sketch vector is a 3-VECTOR (FLOAT ARRAY) whose first two elements are the X and Y sketch coordinates, and whose third element (Z) is 0. The function

```
V = %GETSVEC(M,I)
```

returns the sketch VECTOR of the Ith atom of molecule M. To alter the sketch coordinates of the Ith atom of M, use the procedure

```
CALL %SETSVEC(M,I,V)
```

where V is the new sketch VECTOR (only the first two elements of V are used). The procedure %ADJUSTMOL (see Part 3 below) must be called after altering sketch coordinates with %SETSVEC to ensure proper subsequent display of the molecule.

In certain cases (e.g., for molecules made by procedures rather than MAKE MOLECULE), some or all of the atoms may have no sketch coordinates. Such a condition can be tested with the expression EMPTY(V[1]), where V is a VECTOR resulting from calling %GETSVEC.

Model Coordinates

The model coordinates of an atom define its position in the 3D molecular model. The term model vector refers to a 3-VECTOR whose elements are the model coordinates. The function

```
V = %GETMVEC(M,I)
```

returns the model VECTOR of the Ith atom of molecule M.

The procedure

```
CALL %SETMVEC(M,I,V)
```

sets the model coordinates of the Ith atom of molecule M to the first three elements of the vector V. The procedure %ADJUSTMOL (see Part 3 below) must be called after altering the model coordinates of a molecule using %SETMVEC.

Missing model coordinates can be tested with the expression EMPTY(V[1]) where V is a VECTOR resulting from calling %GETMVEC.

Section 2B: BONDS Table Gatekeepers

The gatekeepers described in this section are used to access the BONDS table of the molecule; they are summarized in Table 3-7. The notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE	
I	FIXED	Bond index
T	TEXT	Bond type ('S','D','T',or 'R')
A1,A2	FIXED	Atom indices

Adding and Deleting Bonds

The procedure %ADDBOND is used to add a bond to a molecule as follows

```
CALL %ADDBOND(M,T,A1,A2,I)
```

where M is the molecule, T is the bond type (a TEXT such as 'S', 'D', 'T' or 'R'), and A1 and A2 are the indices of the atoms at either end of the bond to be added; the FIXED variable I is set to the index of the newly added bond. If A1 and A2 are already bonded, %ADDBOND merely sets the bond type to T and sets I to the index of the existing bond. If A1 = A2 the procedure complains and sets I to 0. Note that both of the end atoms must be defined before calling %ADDBOND; thus, a water molecule could be made by the following code fragment:

```
DECLARE  H2O  MOLECULE;        /* Water molecule */
DECLARE (IH1,IO,IH2) FIXED;  /* Atom indices */
DECLARE (IB1,IB2)  FIXED;    /* Bond indices */
                 ...
CALL %MAKEMOL(H2O);/* Setup structure - see Part 3 */
CALL %ADDATOM(H2O,'H',IH1);  /* Add H */
CALL %ADDATOM(H2O,'O',IO);   /* Add O */
CALL %ADDBOND(H2O,'S',IH1,IO,IB1);  /* H-O bond */
CALL %ADDATOM(H2O,'H',IH2);  /* Add H */
CALL %ADDBOND(H2O,'S',IO,IH2,IB2);  /* O-H bond */
CALL %ADJUSTMOL(H2O);        /* Set internal fields */
                 ...
```

Table 3-7: BONDS TABLE Gatekeeper Procedures

Calling Form	Usage
CALL %ADDBOND(M,T,A1,A2,I)	Adds a bond of type T to mol. M between atoms A1 and A2. I is set to index of new bond.
CALL %DELETEBOND(M,I)	Delete bond I of molecule M, adjust all bond index references in M.
%LASTBOND(M)	Index of last bond in molecule M.
%GETBONDTYPE(M,I)	Returns (TEXT) type of bond I of molecule M ('S', 'D', 'T', or 'R').
CALL %SETBONDTYPE(M,I,T)	Set type of bond I of molecule M to T.
%GETATOM1(M,I)	Returns (FIXED) index of atom at one end of bond I in molecule M.
%GETATOM2(M,I)	Returns (FIXED) index of atom at other end of bond I in molecule M.

Notation Conventions:

M	MOLECULE	To access or modify
I	FIXED	Bond index
T	TEXT	Bond type ('S', 'D', 'T', 'R')
A1,A2	FIXED	Atom indices

The procedure %DELETEBOND is used to delete a bond from a molecule as follows:

```
CALL %DELETEBOND(M,I)
```

where M is the molecule and I is the FIXED index of the bond to be deleted.

Finding and Changing Bond Information

The function

```
I = %LASTBOND(M)
```

returns the FIXED index of the last bond in the molecule M. In general, BONDS table gatekeepers check the validity of their bond index arguments, and abort if the index is less than one or greater than %LASTBOND(M).

Bond Types

The function

```
T = %GETBONDTYPE(M,I)
```

returns a TEXT which is the type of the Ith bond of molecule M (e.g., 'S', 'R', 'D', 'T'). The type of the Ith bond of M can be set to the TEXT T by the procedure

```
CALL %SETBONDTYPE(M,I,T)
```

NOTE: while the data structure per se places no restrictions on bond types, many of the system routines will complain about bond types other than those above.

The functions

```
A1 = %GETATOM1(M,I)
```

and

```
A2 = %GETATOM2(M,I)
```

return the FIXED indices of the two atoms connected to the Ith bond of molecule M. There are no corresponding %SETATOM1 or %SETATOM2 routines, since the procedure %ADDBOND sets the corresponding columns in the BONDS table.

Table 3-8: CONSTRAINTS Table Gatekeeper Procedures

Calling Form	Usage
CALL %ADDCONSTRAINT (M,T,AA,BA,V,I)	Adds constraint of type T, involving atoms AA and bonds BA, with value V, to molecule M. Index of new constraint is stored in I.
CALL %DELETECONSTRAINT(M,I)	Deletes constraint I of molecule M.
%LASTCONSTRAINT(M)	Returns (FIXED) index of last constraint in molecule M.
%GETCONTYPE(M,I)	Returns (TEXT) type of constraint I of molecule M.
CALL %SETCONTYPE(M,I,T)	Sets type of contraint I of molecule M to T.
%GETCONATOMS(M,I)	Returns (FIXED ARRAY) indices of atoms in constraint I of molecule M.
CALL %SETCONATOMS(M,I,AA)	Sets atoms in constraint I of molecule M to AA.
%GETCONBONDS(M,I)	Returns (FIXED ARRAY) indices of bonds in constraint I of molecule M.
CALL %SETCONBONDS(M,I,BA)	Sets bonds in constraint I of M to BA.
%GETCONVALUE(M,I)	Returns (ANY) value of constraint I of M.
CALL %SETCONVALUE(M,I,V)	Sets value of constraint I of molecule M to V.

Notation Conventions:

M	MOLECULE	To access or modify
I	FIXED	Constraint index
T	TEXT	Constraint type
AA	FIXED ARRAY	Indices of constrained atoms
BA	FIXED ARRAY	Indices of constrained bonds
V	ANY	Value of constraint

Section 2C: CONSTRAINTS Table Gatekeepers

This section discusses the gatekeepers which access and modify information in the CONSTRAINTS table of the molecule; they are summarized in Table 3-8. The notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE	
I	FIXED	Constraint index
T	TEXT	Constraint type
AA	FIXED ARRAY	Indices of constrained atoms
BA	FIXED ARRAY	Indices of constrained bonds
V	ANY	Value of constraint

Adding and Deleting Constraints

A new constraint can be added to the molecule M with the procedure

```
CALL %ADDCONSTRAINT(M,T,AA,BA,V,I)
```

where T (a TEXT) is the type of the new constraint, AA and BA (FIXED ARRAYs) are the indices of atoms and bonds involved in the constraint (either or both may be EMPTY), and V is the value of the constraint. I is set to the index of the new constraint by %ADDCONSTRAINT. In the event that no CONSTRAINTS table exists in molecule M yet, one is created before adding the new constraint. The constraint types that are recognized by the PROPHET COMPUTE MODEL command are presented in Table 3-5 in Part 1 of this chapter. %ADDCONSTRAINT does not prevent you from setting constraints of types other than those mentioned in Table 3-5, but other types will simply be ignored when you are computing a model.

The Ith constraint of molecule M can be deleted by the procedure

```
CALL %DELETECONSTRAINT(M,I);
```

Finding and Changing Constraint Information

The function

```
I = %LASTCONSTRAINT(M)
```

returns the FIXED index of the last constraint in the molecule M; however, if M has no CONSTRAINTS table, the returned value is -1. The CONSTRAINTS table is not normally created in the process of constructing a molecule; a table is created, when needed, by %ADDCONSTRAINT.

The CONSTRAINTS table gatekeepers check the validity of their constraint index argument, and abort if it is less than 1 or greater than the value returned by %LASTCONSTRAINT.

The function

```
T = %GETCONTYPE(M,I)
```

returns a TEXT which is the type of the Ith constraint of molecule M. This value can be altered by the procedure

```
CALL %SETCONTYPE(M,I,T)
```

where T is a TEXT which is the new type for the Ith constraint.

The function

```
AA = %GETCONATOMS(M,I)
```

returns a FIXED ARRAY which contains the indices of atoms to which the Ith constraint of molecule M applies. This can be altered by the procedure

```
CALL %SETCONATOMS(M,I,AA)
```

where AA is a FIXED ARRAY of atom indices.

The function

```
BA = %GETCONBONDS(M,I)
```

returns a FIXED ARRAY which contains the indices of the bonds to which the Ith constraint of molecule M applies. The bonds to which a constraint applies may be altered by the procedure

```
CALL %SETCONBONDS(M,I,BA)
```

where BA is a FIXED ARRAY of bond indices.

The function

```
V = %GETCONVALUE(M,I)
```

returns the value of the Ith constraint of molecule M; for example, if the type of constraint I is 'BONDLENGTH', then %GETCONVALUE(M,I) is the length (in Angstroms) to which the specified bond is constrained. The nature of the value returned for each system-recognized constraint type is described in Table 3-5 in Part 1 of this chapter. The value of the Ith constraint of M can be set to V (of any data type) by the procedure

```
CALL %SETCONVALUE(M,I,V);
```

Section 2D: ATTRIBUTES Gatekeepers

This section describes the gatekeepers which access the ATTRIBUTES of a molecule; they are summarized in Table 3-9. The notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE or MULTIMOLECULE	
A	TEXT	Name of an attribute
V	ANY	Value of an attribute

Finding an Attribute

The function

```
V = %GETATTRIBUTE(M,A)
```

returns the value of the attribute named A (a TEXT) of the molecule M. If there is no attribute named A, an EMPTY value of type ANY is returned.

Setting and Deleting Attributes

The value of an attribute named A in molecule M can be set to the value V (of any type) by the procedure

```
CALL %SETATTRIBUTE(M,A,V)
```

It is up to you to determine what types are appropriate as the values of any attributes you wish to define. Notice that setting an ATTRIBUTE's value to EMPTY is roughly equivalent to deleting it, since %GETATTRIBUTE returns EMPTY for attributes that are not found. However, a slightly more efficient means of deleting an attribute (which does not leave a partially empty row in the ATTRIBUTES array) is to use the procedure

```
CALL %DELETEATTRIBUTE(M,A)
```

Table 3-9: ATTRIBUTES Gatekeeper Procedures

Calling Form	Usage
%GETATTRIBUTE(M,A)	Returns the (ANY) value of the attribute named A of M.
CALL %SETATTRIBUTE(M,A,V)	Sets the value of the attribute named A of M to V.
CALL %DELETEATTRIBUTE(M,A)	Deletes the attribute named A from M.

Notation Conventions:

M	MOLECULE or MULTIMOLECULE	Containing the ATTRIBUTES
A	TEXT	Name of an attribute
V	ANY	Value of an attribute

Section 2E: Miscellaneous Gatekeepers

This section describes gatekeepers which interface to several of the subfields in the molecule; they are summarized in Table 3-10. Some of the notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE	
A,A1,A2	FIXED	Atom indices
B	FIXED	Bond index
TF	BOOL	

FLAGS

The function

```
TF = %GETFLAG(M,F)
```

where F is one of the symbolic flag names (%M%FRAG, %M%MDL, %M%SKT, %M%DSK, or %M%UNBD), returns the BOOLean value of the corresponding flag; the interpretation of these flags is described in Chapter II. To set the value of a flag F of molecule M to the BOOL value B, use the procedure

```
CALL %SETFLAG(M,F,B)
```

WARNING!

You should *never* alter the value of a flag in an existing molecule. Let %ADJUSTMOL keep the flags set to the proper values. About the only occasion to use %SETFLAG is when you have an EMPTY molecule that you wish to make as a disk molecule. In that case, you may call %SETFLAG to set %M%DSK to TRUE *before* calling %MAKEMOL.

Bonded Atoms

The function

```
B = %BONDBETWEEN(M,A1,A2)
```

Table 3-10: Miscellaneous Molecule Gatekeeper Procedures

Calling Form	Usage
%GETFLAG(M,F)	Returns (BOOL) value of flag F.
CALL %SETFLAG(M,F,TF)	Sets value of flag F of molecule M to TF (TRUE or FALSE).
%BONDBETWEEN(M,A1,A2)	Returns (FIXED) index of bond between atoms A1 and A2 in molecule M, or 0 if they are not bonded.
%ATOMNEIGHBORS(M,A)	Returns (FIXED ARRAY) indices of atoms bonded to atom A of molecule M.
%OTHERATOM(M,B,A)	Returns (FIXED) index of atom at end of bond B opposite atom A in molecule M, or 0 if A is not connected to B.
%SYSTEMH(M,A)	Returns (BOOL) TRUE if atom A of molecule M is a system-added hydrogen (no sketch coordinates).

Notation Conventions:

M	MOLECULE	To access or modify
F	BIT	Symbolic flag name: %M%MDL, %M%SKT, %M%FRAG %M%DSK, or %M%UNBD
TF	BOOL	Flag value (TRUE or FALSE)
A1,A2	FIXED	Atom indices
A	FIXED	Atom index
B	FIXED	Bond index

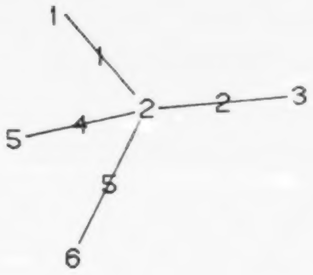
where A1 and A2 are FIXED indices of atoms in the molecule M, returns the FIXED index of the bond between atoms A1 and A2. If the two specified atoms are not bonded, 0 is returned.

The function

AA = %ATOMNEIGHBORS(M,I)

returns a FIXED ARRAY which contains the indices of atoms bonded to the Ith atom of molecule M. It is guaranteed that the resulting array is in the same order as the one produced by %GETBONDA, thus:

%ATOMNEIGHBORS(M,2) ==> [1,3,5,6]



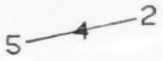
%GETBONDA(M,2) ==> [1,2,4,5]

The function

A2 = %OTHERATOM(M,B,A)

returns the FIXED index of an atom which is connected to bond B of molecule M at the end opposite to atom A, viz.:

%OTHERATOM(M, 4, 5) ==> 2



The result of %OTHERATOM is meaningless when atom A is not attached to bond B in M; in that case, 0 is returned. NOTE: the result of %OTHERATOM(M,B,A) is emphatically not the same as %OTHERATOM(M,A,B) (the degree "MBA" is a good mnemonic).

System-
Added
Hydrogens

The function

```
TF = %SYSTEMH(M,I)
```

returns TRUE if the Ith atom of molecule M is a "system-added hydrogen", i.e., a hydrogen introduced by the MAKE MOLECULE or %HBALANCE quiz to balance valences of C's and N's, rather than explicitly sketched in by the user. %SYSTEMH assumes that any atom of type 'H' without sketch coordinates is system-added.

Part 3: Other Molecule Procedures

This part describes other procedures for manipulating and accessing the molecule data structure, at a somewhat higher level than the gatekeepers described in Part 2. These procedures are grouped into eight sections as follows:

Section 3A	Procedures that Make and Adjust Molecules
Section 3B	Procedures that Retrieve Chemical Information from System Tables
Section 3C	Procedures that Compute Properties of a Molecule
Section 3D	Procedures that Manipulate Molecular Models
Section 3E	Procedures that Access and Set Constraints
Section 3F	Procedures Used in Substructure Search
Section 3G	Procedures that Handle Empirical Chemical Formulas
Section 3H	Procedures Used to Mark and Unmark Parts of Molecules

Section 3A: Making and Adjusting Molecules

The procedures described in this section provide you with tools for creating new molecules, for copying molecules, for making molecules that consist of parts of preexisting molecules, and for determining if molecules are to be disk-based or core-based. In addition, we describe a procedure, %ADJUSTMOL, that should always be called whenever basic changes have been made to a molecule. The procedures described in this section are summarized in Table 3-11. The notation conventions for the variable names used in several sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE	previously existing
NEWM	MOLECULE	being made

Making a Molecule

A new molecule can be created using the procedure

```
CALL %MAKEMOL(NEWM);
```

where NEWM is a variable previously declared to be of type MOLECULE. As a result of the call, the structure of the ATOMS and BONDS tables is established (column types and headings), and various other subfields are set to initial values. As described above, the CONSTRAINTS table and the ATTRIBUTES array are created only when needed, since they may not appear in all molecules.

Adjusting a Molecule After Changing It

If a user program manipulates any information in the ATOMS or BONDS tables, it is necessary to update certain non-table subfields of the molecule with the procedure

```
CALL %ADJUSTMOL(M)
```

where M is the molecule that was altered. Failure to call %ADJUSTMOL may result in certain anomalies in

Table 3-11: Procedures that Make and Adjust Molecules

Calling Form	Usage
CALL %ADJUSTMOL(M)	Updates system information in M; should be called after manipulation of coordinates, atom types, etc.
CALL %CONVERT2COREMOL(M)	Converts M to a core-based molecule.
CALL %CONVERT2DISKMOL(M)	Converts M to a disk-based molecule.
CALL %MAKEMOL(M)	Initializes data structure of molecule M. M must be previously declared to be of type MOLECULE.
%MAKEMOLSELECT(M,ATMS, BNDS,DSKCOR)	Returns a MOLECULE constructed from the specified atoms and bonds in M.
%MOLCOPY(M,DSKCOR)	Returns a MOLECULE that is a copy of M, either disk-based or core-based, as specified.

Notation Conventions

M	MOLECULE	
ATMS	FIXED or TEXT or FIXED ARRAY or BOOL ARRAY	Specification of atoms to be included
BNDS	FIXED or TEXT or FIXED ARRAY or BOOL ARRAY	Specification of bonds to be included
DSKCOR	TEXT	Indication of type of new molecule, 'DISK' or 'CORE'

trying to display the molecule. %ADJUSTMOL sets various internal subfields of the molecule and sets the flags %M%MDL and %M%SKT to TRUE if any model or sketch coordinates are encountered.

Making a Molecule from an Existing Molecule

To make a new molecule that is an exact copy of an existing molecule, use the function %MOLCOPY in a statement like the following:

```
NEWM = %MOLCOPY(M, MTYPE)
```

MTYPE in this example is a TEXT variable that has been set to 'DISK' if NEWM is to be a disk molecule or 'CORE' if NEWM is to be a normal core molecule. If you simply want the new molecule to be the same type as the old one, use a statement like:

```
NEWM = %MOLCOPY(M, )
```

Note that using the function COPY is not sufficient when you are dealing with disk molecules; %MOLCOPY _must_ be used to get the disk-based tables copied properly.

To make a new molecule that consists of some part of an existing molecule, you can use the function %MAKEMOLSELECT with a statement such as:

```
NEWM = %MAKEMOLSELECT(M, ATMS, BNDS, MTYPE)
```

where MTYPE is a TEXT variable treated in the same way as described for %MOLCOPY above. ATMS and BNDS are variables that describe which atoms and bonds are to be included in the new molecule. They may be of type FIXED, TEXT, FIXED ARRAY or BOOL ARRAY. The interpretations of these different data types are described in the following detailed points about how %MAKEMOLSELECT works:

1. If ATMS or BNDS is of type FIXED, the value indicates the column number of a selecting column in the original molecule's ATOMS or BONDS table (possibly set up by the %MARKATOMS or %MARKBONDS procedures described below).

2. If ATMS or BNDS is of type TEXT, the text string indicates the column heading of a selecting column in the original molecule's ATOMS or BONDS table.

3. If ATMS or BNDS is of type FIXED ARRAY, the values indicate the indices of the atoms or bonds are to be included in the new molecule.

4. If ATMS or BNDS is of type BOOL ARRAY, the members of the array indicate whether the corresponding indices of the atoms or bonds are to be included in the new molecule. Those to be included should be TRUE, and all other positions should be EMPTY or FALSE.

5. The ATMS and BNDS arguments need not be of the same data type. In addition, either ATMS or BNDS may be EMPTY in your call to %MAKEMOLSELECT.

6. The new molecule's atoms consist of those atoms that are explicitly selected by ATMS plus any connected by bonds explicitly selected by BNDS.

7. The new molecule's bonds consist of those bonds that are explicitly selected by BNDS plus any that connect two atoms explicitly selected by ATMS.

8. All sketch and model coordinates found in the original molecule are copied over for those atoms that are included in the new molecule.

9. The carbon and nitrogen atoms in the new molecule are always hydrogen balanced by the system.

10. A constraint in the old molecule is copied to the new molecule if and only if all the atoms and bonds referenced by the constraint are included in the new molecule.

11. Extra columns in the ATOMS or BONDS tables of the old molecule (such as WIPKE columns, other selecting columns, or columns used for labeled displays) are copied over to the new molecule.

12. Other subfields of the old molecule are also copied over, including the NAMES, ATTRIBUTES, NOTES and %TRANS subfields.

13. Note that %MAKEMOLSELECT works in exactly the same fashion as the MAKE MOLECULE...FROM...SELECTING command described in Chapter II, except that %MAKEMOLSELECT does not invoke the MAKE MOLECULE quiz when it is done.

Converting Molecules

In Chapter II we described how the ATOMS, BONDS and (optional) CONSTRAINTS tables in most molecules are core-based; that is, they are kept right in your workarea when you are working with a molecule. When a molecule becomes quite large (several hundred atoms) however, it may no longer be possible to keep these tables in your workarea and still have room to perform any useful operations. Hence such molecules are most conveniently handled when their tables are disk-based, at a certain cost in speed of handling.

Normally, you know when you first construct a molecule whether it will be large enough to justify making it disk-based. There may be times when you want to convert a normal core-based molecule to a disk-based molecule, for example, if the molecule has turned out to be larger than you originally anticipated. This can be accomplished with a statement of the form:

```
CALL %CONVERT2DISKMOL(M)
```

where M is the molecule to be converted. Similarly, to convert a disk-based molecule to a core-based molecule, use the procedure

```
CALL %CONVERT2COREMOL(M)
```

where once again M is the molecule that you wish to convert. If the molecule given as the argument to either one of these procedures is already of the type that you intended to convert it to, these procedures do nothing to it.

Section 3B: Retrieving Chemical Information from System Tables

PROPHET has an internal system table that contains the chemical information it uses to build three-dimensional models of molecules. While you cannot access this table directly, the information in it is available in three public tables that are reproduced in Appendix A of this manual. In addition, system procedures described in this section enable you to retrieve specific items of information from the internal table. These procedures are summarized in Table 3-12.

System Element Information

Information about any element (atom type) that PROPHET knows about can be retrieved with the procedure

```
CALL %FINDELEMENT(AT, NAME, WT, COVRAD, VDWRAD);
```

where AT is a TEXT containing a valid atom type (e.g., 'C', 'H', 'O', or some other atomic symbol). As a result of the call, the other variables are set as follows:

Variable Name	Data Type	
NAME	TEXT	Full element name (e.g.,'HYDROGEN')
WT	FLOAT	Atomic weight in atomic mass units
COVRAD	FLOAT	Covalent radius in angstroms
VDWRAD	FLOAT	van der Waals radius in angstroms

If the atom type AT is not known to PROPHET, EMPTYs are assigned to the remaining four arguments. PROPHET maintains information about covalent and van der Waals radii only for the most common elements; hence, if you call %FINDELEMENT for a less common atom type, EMPTYs will be assigned to COVRAD and VDWRAD.

If you require only certain data, you can omit some of the variables. For example, to retrieve only the atomic weight of hydrogen in the FLOAT variable HWT, the following statement could be used:

```
CALL %FINDELEMENT('H', , HWT, ,);
```

Table 3-12: Procedures That Retrieve Chemical Information from System Tables

Calling Form	Usage
%FINDBONDLENGTH(AT1,AT2,BT)	Returns (FLOAT) length of BT-type bond between atom types A1 and A2 in angstroms.
CALL %FINDELEMENT(AT,NM,AW, CR,VR)	Searches element table for atom type AT, sets NM to full (TEXT) name, AW to (FLOAT) atomic weight, CR to (FLOAT) covalent radius, and VR to (FLOAT) van der Waals radius. Missing values are set to EMPTY.
%GETRINGMOL(DICTYPE,ROWIND)	Returns MOLECULE stored in row specified by ROWIND in ring dictionary specified by DICTYPE.

Notation Conventions

AT	TEXT	Element symbol (e.g., 'C')
AT1,AT2	TEXT	Element symbols
AW	FLOAT	Atomic weight
BT	TEXT	Bond type('S','D','T','R')
CR	FLOAT	Covalent radius
DICTYPE	TEXT	'PRIVATE' or 'SHARABLE' or 'PUBLIC'
NM	TEXT	Full element name
ROWIND	FIXED or TEXT	Row index or row name in ring dictionary
VR	FLOAT	Van der Waals radius

System Bond Information

The function

```
L = %FINDBONDLENGTH(AT1, AT2, BT)
```

returns a FLOAT length (in angstroms) for a bond of the type specified by BT (a TEXT) between atoms of types AT1 and AT2 (TEXTs). For example,

```
L = %FINDBONDLENGTH('C','O','D')
```

sets L to the value 1.23 -- the length in angstroms of a Double bond between Carbon and Oxygen.

PROPHET knows standard bond lengths for a number of common atom/bond/atom combinations, as illustrated in the public table BONDLENGTHS in Appendix A. For atom/bond/atom combinations not in this table, %FINDBONDLENGTH first tries to retrieve covalent radii for the two atom types via %FINDELEMENT. If both atom types have stored covalent radii, the two radii are added to get a standard bond length. If one or both radii are missing, %FINDBONDLENGTH returns EMPTY.

Retrieving Ring Models From Ring Dictionaries

You can retrieve a standard ring molecule from one of the three ring dictionaries to which you have access (PRIVATE, SHARABLE or PUBLIC, as described in Chapter II, Part 2). Use the function %GETRINGMOL, which you call with a statement such as the following:

```
M = %GETRINGMOL (DICTYPE, ROWIND)
```

DICTYPE is a variable of type TEXT set to 'PRIVATE', 'SHARABLE' or 'PUBLIC'. ROWIND may be a FIXED row number in the appropriate ring dictionary or the TEXT name of the row. %GETRINGMOL returns the MOLECULE stored in that row. If, for example, you wish to examine the standard benzene ring kept in PUBLIC RINGS, you could retrieve it with the following statement:

```
BENZENE = %GETRINGMOL ('PUBLIC', 'BENZENE')
```

Section 3C: Computing Properties of a Molecule

A number of properties of a molecule, such as the molecule's total atomic weight, can be computed from the basic information stored in the molecule's ATOMS and BONDS tables. The PROPHET procedures that are provided to help you compute such properties are described in this section. These procedures are also summarized in Table 3-13.

Per Cent Composition

The function

```
COMP = %COMPOSITION(M)
```

returns a TEXT which describes the composition of the molecule M. The format of the resulting TEXT is best described by example. If you type the command

```
TYPE %COMPOSITION(ACETONE)
```

PROPHET types the following:

```
CARBON          62.04%
HYDROGEN        10.41%
OXYGEN          27.55%

TOTAL WEIGHT    58.08
```

Typing Molecule Information

The procedure

```
CALL %TYPEMOL(M,OPT);
```

causes information about molecule M to be typed out; the format is controlled by the TEXT argument OPT, which can take on the following values:

OPT	What is typed
MIN	Number of atoms, bonds, formula, names
CONN	Connectivity table for the molecule
%COMP	Composition (%-weight) for molecule
ALL	Everything typed by 'MIN', 'CONN', and '%COMP' options

Table 3-13: Procedures that Compute Properties of a Molecule

Calling Form	Usage
%ADIHEDRAL(M,A1,A2,A3,A4)	Returns (FLOAT) dihedral angle formed by four atoms A1, A2, A3 and A4 in M.
%ATOMSDIST(M,A1,A2)	Returns (FLOAT) distance between Atoms A1 and A2 in M.
%BDIHEDRAL(M,B1,B2,B3)	Returns (FLOAT) dihedral angle formed by three adjacent bonds B1, B2 and B3 in M.
%BONDEDCOUNT(M,A1)	Returns (FLOAT) number of valence slots filled in atom A1 of M.
%COMPOSITION(M)	Returns (TEXT) describing composition of M.
%COMPUTEATOMSANGLE(M,A1, A2,A3)	Returns (FLOAT) angle formed by three atoms A1, A2 and A3 in M.
CALL %CONNT2A(M,CONNA)	Sets CONNA to connection table for M.
%FINDBONDANGLE(M,B1,B2)	Returns (FLOAT) angle between two adjacent bonds B1 and B2 in M.
%MOLBONDLENGTH(M,B1)	Returns (FLOAT) length of bond B1 in M.
CALL %MOLCOORDS(M,A1,X,Y)	Sets X and Y to (FLOAT) screen coordinates of atom A1 in M. M must be currently displayed.
%MOLWEIGHT(M)	Returns (FLOAT) molecular weight of M.
CALL %TYPEMOL(M,OPT)	Types information about M, controlled by OPT.
CALL %VERIFYMOLCONNS(M)	Aborts with appropriate error message if any inconsistencies are found in M.

Notation Conventions

M	MOLECULE	
A1,A2,A3,A4	FIXED	Atom indices
B1,B2,B3	FIXED	Bond indices
CONNA	ANY ARRAY ARRAY	Array to hold connection table of a molecule
OPT	TEXT	Option for %TYPEMOL: 'MIN','CONN', '%COMP','ALL'
X,Y	FLOAT	Screen coordinates

For example, if you typed the command

```
CALL %TYPEMOL(ACETONE,'ALL')
```

PROPHET would type the following:

```
10 ATOMS, 9 BONDS

                FORMULA: C3 H6 O


CARBON          62.04%
HYDROGEN        10.41%
OXYGEN          27.55%

TOTAL WEIGHT    58.08

INDEX     ATOM     CONNECTIONS          BONDS
  1        C       2  8  9  10     S  S  S  S
  1        C       1  3  4         S  S  D
  3        C       2  5  6   7     S  S  S  S
  4        O       2               D
  5        H       3               S
  6        H       3               S
  7        H       3               S
  8        H       1               S
  9        H       1               S
 10        H       1               S
```

Computing Molecular Weights

The function

```
WT = %MOLWEIGHT(M)
```

returns the FLOAT molecular weight of molecule M. %MOLWEIGHT may also be called with an argument of type CHEMTEXT containing the empirical formula of the compound whose molecular weight you would like to compute. For example, you might use the following sequence of commands to get the molecular weight of acetone without actually making the molecule:

```
MAKE CHEMTEXT CT<GO>
ENTER CHEMTEXT FORMULA?C3 H6 O<GO>
TYPE %MOLWEIGHT(CT)<GO>
58.081
```

Computing Bond Angles

To compute the angle between two adjacent bonds in a molecule, use a statement like:

```
ANG = %FINDBONDANGLE(M, B1, B2)
```

where B1 and B2 are the FIXED indices of two adjacent bonds in molecule M. %FINDBONDANGLE returns the FLOAT angle in degrees between the two specified bonds; in the example above, this FLOAT value would be assigned to the variable ANG.

Alternatively, you can compute an angle between two vectors in a molecule with a statement like:

```
ANG = %COMPUTEATOMSANGLE(M, A1, A2, A3)
```

where A1, A2 and A3 are the FIXED indices of three different atoms in molecule M. The FLOAT value returned by %COMPUTEATOMSANGLE (and assigned to the variable ANG in this example) is the angle in degrees between a vector from atom A1 to atom A2 and a vector from atom A2 to atom A3. If atom A1 is bonded to atom A2 and atom A2 is bonded to atom A3, the value returned is the bond angle between the bond joining atom A1 to atom A2 and the bond joining atom A2 to atom A3.

Computing Distances

You can determine the length of a particular bond in a molecule with a statement like the following:

```
L = %MOLBONDLENGTH(M, B1)
```

where B1 is the FIXED index of the bond in molecule M whose length you wish to measure. The FLOAT value returned by %MOLBONDLENGTH (and assigned to L in this example) is the length in angstroms of the specified bond.

You can determine the distance between any two atoms in a molecule, whether they are bonded or not, with a statement like the following:

```
L = %ATOMSDIST(M, A1, A2)
```

where A1 and A2 are the FIXED indices of two atoms in molecule M. The FLOAT value returned by %ATOMDIST (and assigned to the variable L in this example) is the distance in angstroms between the two specified atoms.

Computing Dihedral Angles

To compute the dihedral angle (torsion angle) formed by three adjacent bonds in a molecule, use a statement like the following:

```
ANG = %BDIHEDRAL(M, B1, B2, B3)
```

where B1, B2 and B3 are the FIXED indices of the three adjacent bonds in atom M. A FLOAT value is returned by %BDIHEDRAL (and assigned to the variable ANG in this example). It is the dihedral angle in degrees between bonds B1 and B3 measured clockwise when looking down bond B2 from bond B1 toward bond B3, as illustrated in the Newman projection in Figure 3-4.

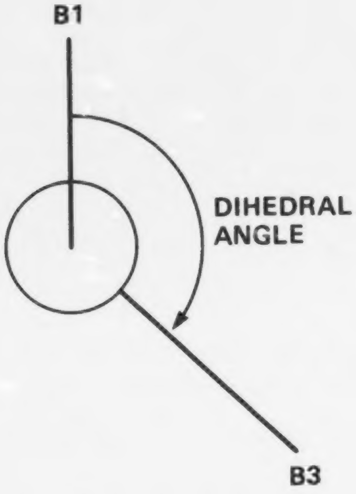


Figure 3-4. Clockwise dihedral angle formed by bonds B1 and B3 when looking down the axis of bond B2.

To compute the dihedral angle formed by any four atoms in a molecule, use a statement like the following:

```
ANG = %ADIHEDRAL(M, A1, A2, A3, A4)
```

where A1, A2, A3 and A4 are the FIXED indices of four atoms in molecule M. The FLOAT value returned by %ADIHEDRAL (and assigned to variable ANG in this example) is the torsion angle in degrees between the vector from atom A1 to atom A2 and the vector from atom

A3 to atom A4 measured clockwise about the vector from atom A2 to atom A3. The four atoms need not necessarily be bonded to one another. But, if atom A1 is bonded to atom A2, and atom A2 is bonded to Atom A3, and atom A3 is bonded to atom A4, the angle returned by %ADIHEDRAL is the same dihedral angle returned by %BDIHEDRAL given the three connecting bonds as arguments.

Checking Valences

To determine the number of valence "slots" that are filled for a particular atom, use a statement like the following:

```
V = %BONDEDCOUNT(M, A1)
```

where A1 is the FIXED index of the atom in molecule M in which you are interested. The FLOAT value returned by %BONDEDCOUNT (and assigned to the variable V in this example) is the number of valence slots that are filled for the specified atom. %BONDEDCOUNT adds 1.0 to the returned value for each single bond connected to atom A1, it adds 2.0 for each double bond, and it adds 3.0 for each triple bond. If atom A1 has any resonant bonds attached to it, %BONDEDCOUNT adds 1.0 plus the number of resonant bonds to the value it returns. %BONDEDCOUNT does not check whether the nominal valence of the atom in question matches the number of valence slots actually filled.

Getting the Connection Table

The procedure

```
CALL %CONNT2A(M, CONNA)
```

uses the structure of molecule M to create a connection "table" in the ANY ARRAY ARRAY specified as the second argument (CONNA in this example). There is one "row" for each atom in M. Each "row" of CONNA has the structure:

CONNA[I,1]	Atom Type <TEXT>
CONNA[I,2]	Number of connections <FIXED> = N
CONNA[I,3]	Index of neighbor Atom 1 <FIXED>
..	...
CONNA[I,N+2]	Index of neighbor Atom N <FIXED>
CONNA[I,N+3]	Type of bond to neighbor Atom 1 <TEXT>
..	...
CONNA[I,2N+2]	Type of bond to neighbor Atom N <TEXT>

Verifying Connection Tables

To check that every reference to the BONDS table of a molecule from the molecule's ATOMS table is consistent with every reference to the ATOMS table from the BONDS table, as well as to make sure that the molecule has no disconnected pieces, use a call like the following:

```
CALL %VERIFYMOLCONNS(M)
```

where M is the molecule you wish to check. If %VERIFYMOLCONNS finds anything wrong with the ATOMS or BONDS table, it aborts with appropriate error messages. Suppose, for example, you have a molecule with the following inconsistent ATOMS and BONDS tables.

ATOMS: 3R X 7C

	1 TYPE	2 BOND ARRAY	3 SKETCH X	4 SKETCH Y	5 MODEL X	6 MODEL Y	7 MODEL Z
1	C	[10]	4.80	-5.98			
2	C	[1,2,3]	6.70	-4.92			
3	C	[2,4,5,6]	8.10	-5.94			

BONDS: 6R X 3X

	1 TYPE	2 ATOM 1	3 ATOM 2
1	S	1	2
2	S	2	3
3	D	2	4
4	S	3	5
5	S	3	6
6	S	3	7

If you called %VERIFYMOLCONNS with this molecule as its argument, %VERIFYMOLCONNS would type out the following messages:

```
ATOM 1 HAS INCORRECT BOND ARRAY
BOND 1 HAS INCORRECT ATOM CONNECTIONS
BOND 3 HAS INCORRECT ATOM CONNECTIONS
BOND 4 HAS INCORRECT ATOM CONNECTIONS
BOND 5 HAS INCORRECT ATOM CONNECTIONS
BOND 6 HAS INCORRECT ATOM CONNECTIONS
MOLECULE INVALID

IN %VERIFYMOLCONNS
ABORTED
```

Screen Coordinates of an Atom

The procedure %MOLCOORDS gives you limited access to the PROPHET molecule display system. If the molecule M currently appears on a display terminal screen, the coordinates (in screen units) of the Ith atom can be retrieved with the statement

```
CALL %MOLCOORDS(M, I, X, Y)
```

If the atom with FIXED index I in molecule M is displayed, then the FLOAT variables X and Y are set to the screen coordinates at which it appears; these can be used, in turn, in a call to such routines as %DTEXT or %DBOX (see the PL/PROPHET User's Guide). If M is not currently displayed, or if atom I is not shown, then X and Y are set to EMPTY.

Section 3D: Manipulating Molecular Models

Once a three-dimensional model has been computed for a molecule or three-dimensional coordinates have been supplied by you, there are a number of procedures available that will enable you to make major or minor modifications to the model. These procedures are summarized in Table 3-14.

Filling in Bonds

%CONNECTMOL is an interactive procedure that helps you fill in the proper bonds in a molecule you are building from a set of atomic coordinates. The public procedures CRYST and CONNECT (see Chapter V) both call %CONNECTMOL. In other situations, you can call %CONNECTMOL directly with a statement like

```
CALL %CONNECTMOL(M)
```

where M is a molecule in a state where all the atoms have already been assigned model coordinates, but where no bonding information has yet been set. When you call %CONNECTMOL for an appropriately constructed molecule, it does three things:

1. First, %CONNECTMOL adds bonds of various types to the molecule according to the proximity of atoms and your responses.
2. It then adjusts the sketch and model of the molecule.
3. Finally, it allows you to modify the molecule. You can add, delete, and change bonds, and add hydrogens to balance carbon atoms.

Adding Bonds in %CONNECTMOL

In the first step, %CONNECTMOL bonds atoms if they are close enough together. ATOMI and ATOMJ are bonded if their interatomic distance is less than the elemental radii of ATOMI plus ATOMJ plus a tolerance (which has a default value of .3 angstroms). However, you have great flexibility in affecting how atoms will be bonded.

Table 3-14: Procedures that Manipulate Molecular Models

Calling Form	Usage
CALL %CONNECTMOL(M)	Interactively determines types and locations of bonds in M containing model coordinates.
%GETNEWMVEC(M,A1,A2,A3,DIST, BANGLE,TANGLE)	Returns (VECTOR) coordinates of new atom in M given known atoms A1, A2 and A3.
CALL %HBALANCE(M)	Adds hydrogen atoms to balance valences of carbon and nitrogen atoms in M.
CALL %MERGEMOL(M1,M2,SKTXFM, MDLXFM,A1,A2,BT)	Adds M2 to M1, bonding atom A1 in M1 to atom A2 in M2 with a bond of type BT, transforming M2's sketch coordinates by SKTXFM and its model coordinates by MDLXFM.
CALL %TWISTMOL(M,A1,A2,ANG)	Twists model coordinates of M by rotating part of model on A2 end of bond between atoms A1 and A2 by ANG degrees.

Notation Conventions

M,M1,M2	MOLECULE	
A1,A2,A3	FIXED	Atom indices
ANG	FLOAT	Angle in degrees
BANGLE	FLOAT	Bond angle (degrees)
BT	TEXT	Bond type('S','D','T','R')
DIST	FLOAT	Distance (angstroms)
MDLXFM	VECTOR ARRAY	Transformer to apply to model coordinates
SKTXFM	VECTOR ARRAY	Transformer to apply to sketch coordinates
TANGLE	FLOAT	Torsion angle (degrees)

Before any bondings are made, %CONNECTMOL lets you view the elemental radii table, change any value in it, and add new entries to it. The radius of any atom not found in the elemental radii table is assigned a default radius of 1.0 angstrom. You can change this default value if desired. In addition, you can change the tolerance value.

Next, %CONNECTMOL examines each atom pair and bonds the atoms if they are close enough. You can then have C-C, C-O, and C-N bond types identified automatically. If desired, you can specify the interatomic distances that determine the type of bonds to be set. Otherwise, %CONNECTMOL sets bond types as follows:

TYPE OF ATOM I	TYPE OF ATOM J	INTERATOM DISTANCE	BOND TYPE
C	C	>1.44	S
C	C	<1.25	T
C	C	otherwise	D
C	O	>1.27	S
C	O	≤1.27	D
C	N	>1.34	S
C	N	<1.22	T
C	N	otherwise	D

If you do not want any special bond types, %CONNECTMOL adds only single bonds.

If you desire that any rings be made resonant, CONNECTMOL presents each ring one at a time, (by displaying the atom names involved in that ring) and asks whether it should be set to a resonant ring. If any ring is made resonant, all bonds in that ring are set to type R.

At this point, %CONNECTMOL lists a connection table of the molecule constructed thus far. For each atom in the molecule, the index, type, name, neighbors and bondtypes are listed.

%CONNECTMOL then enters an interactive quiz which allows you to add, change or delete bonds and to balance carbon atom valences with hydrogens.

There are two forms of this quiz: one for display terminals, the other for non-display terminals. In the display interaction, you can initiate commands by tablet indication or by typing any proper prefix of a command. The light buttons are shown in Figure 3-9. (One feature not evident from the quiz is that all carbon atoms may be balanced with hydrogens if, after indicating the ADDH command, you type "ALL"). You can receive help by pointing to the HELP command and then the command in question. In the non-display form of the quiz, you must type each command.

After leaving the interactive quiz, %CONNECTMOL checks for fragmented molecules and valence errors. If any problems of this nature are detected, you are allowed to further modify your molecule. The following example illustrates the use of %CONNECTMOL.

%CONNECTMOL Example

In this example, we use coordinate data from the table MOLDATA (Figure 3-5) to make a molecule MOLQ by simply adding atoms with the following commands:

```
DCL MOLQ MOLECULE<GO>
DCL (K,I) FIXED<GO>
DCL (ATOMTYPE,ATOMID) TEXT<GO>
DCL MODCOOR VECTOR<GO>
%MAKEMOL(MOLQ)<GO>
%SETUPATOMNAMES(MOLQ)<GO>
DO I=1 TO %LASTROW(MOLDATA);
ATOMTYPE=%GETCELL(MOLDATA,I,1);
ATOMID=%GETCELL(MOLDATA,I,2);
%R2A(MOLDATA,I,3,5,MODCOOR);
K=%ADDNEWATOM(MOLQ,ATOMTYPE,ATOMID,MODCOOR);
END;<GO>
MOLQ.NAMES[1]='TEST EXAMPLE'<GO>
```

The molecule constructed by these commands is shown in Figure 3-6.

At this point, call %CONNECTMOL with the command:

```
CALL %CONNECTMOL(MOLQ)<GO>
```

The subsequent interaction is illustrated in Figure 3-7. Next, %CONNECTMOL displays a list of the bonds it has found as shown in Figure 3-8. Finally, the molecule is displayed on the screen, complete with bonds and a series of light buttons that allow you to make any changes you wish as shown in Figure 3-9. For the purposes of this example, let us assume that you merely pointed to the EXIT light button. MOLQ would then appear as shown in Figure 3-10.

MOLDATA 25R X 5C

	1 ATOM TYPE	2 ATOM NAME	3 X	4 Y	5 Z
1	C	C1	4.964	0.	1.452
2	C	C2	3.424	0.	1.452
3	C	C3	2.91	0.	0.
4	C	C4	1.37	0.	0.
5	C	C5	2.91	1.257	2.177
6	O	O6	3.642	2.113	2.672
7	C	C7	.685	1.19	0.
8	C	C8	-.685	1.19	0.
9	C	C9	-1.37	0.	0.
10	C	C10	-.685	-1.19	0.
11	C	C11	.685	-1.19	0.
12	O	O12	-2.8	0.	0.
13	H	H13	-3.109	0.	.951
14	O	O14	1.48	1.257	2.177
15	H	H15	1.171	2.081	2.653
16	H	H16	5.331	0.	2.489
17	H	H17	5.331	.898	.933
18	H	H18	5.331	-.898	.933
19	H	H19	3.057	-.898	1.97
20	H	H20	3.276	-.898	-.519
21	H	H21	3.276	.898	-.519
22	H	H22	1.234	2.143	0.
23	H	H23	-1.234	2.143	0.
24	H	H24	-1.234	-2.143	0.
25	H	H25	1.234	-2.143	0.

Figure 3-5. MOLDATA table used in %CONNECTMOL example.

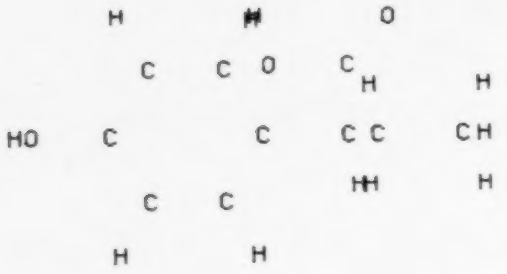


Figure 3-6. MOLQ constructed in %CONNECTMOL example, displayed THREED COMPLETE <u>before</u> calling %CONNECTMOL.

```
THE CONNECTIONS TABLE FOR TEST EXAMPLE IS

1    C    C1       2 16 17 18       S S S S
2    C    C2       1 3 5 19         S S S S
3    C    C3       2 4 20 21        S S S S
4    C    C4       3 7 11           S R R
5    C    C5       2 6 14           S D S
6    O    O6       5                D
7    C    C7       4 8 22           R R S
8    C    C8       7 9 23           R R S
9    C    C9       8 10 12          R R S
10   C    C10      9 11 24          R R S
11   C    C11      4 10 25          R R S
12   O    O12      9 13             S S
13   H    H13      12               S
14   O    O14      5 15             S S
15   H    H15      14               S
16   H    H16      1                S
17   H    H17      1                S
18   H    H18      1                S
19   H    H19      2                S
20   H    H20      3                S
21   H    H21      3                S
22   H    H22      7                S
23   H    H23      8                S
24   H    H24      10               S
25   H    H25      11               S
```

Figure 3-8. %CONNECTMOL summary of connections found in example (Figure 3-7 on reverse).

```
ATOMI AND ATOMJ OF TEST EXAMPLE WILL
BE BONDED IF

        D(I,J) <= R(I) + R(J) + TOL

WHERE D(I,J) IS THE DISTANCE (IN ANGSTROMS) BETWEEN ATOMI AND ATOMJ,
R(I) AND R(J) ARE THE ELEMENTAL RADII OF ATOMI AND ATOMJ
RESPECTIVELY, AND TOL IS THE TOLERANCE
WHICH HAS A DEFAULT VALUE OF .3 ANGSTROMS.
DO YOU WISH TO SEE THE ELEMENTAL RADII TABLE?Y<
THE ELEMENTAL RADII ARE:

   H         BE         B          C          N          O          F
  .300     1.060      .880       .770       .700       .660       .640

   NA        MG         AL         SI         P          S          CL
 1.570     1.400     1.260      1.170      1.100      1.040       .990

   CU        ZN         GA         GE         AS         SE         BR
 1.350     1.310     1.260      1.220      1.180      1.140      1.110

   AG        CD         IN         SN         SB         TE         I
 1.520     1.480     1.440      1.400      1.360      1.320      1.280

DO YOU WISH TO CHANGE ANY VALUE IN THE ELEMENTAL RADII TABLE?N<
DO YOU WISH TO ADD ANY NEW ENTRIES
TO THE ELEMENTAL RADII TABLE?N<
THE ATOMIC RADIUS FOR ALL OTHER ATOMS WILL DEFAULT TO 1.000 ANGSTROM
DO YOU WISH TO CHANGE THIS DEFAULT VALUE?N<
THE TOLERANCE IS 0.3000 ANGSTROMS. DO YOU WISH TO CHANGE IT?N<
DO YOU WISH TO HAVE C-C, C-O, AND C-N
BOND TYPES IDENTIFIED AUTOMATICALLY?Y<
DO YOU WISH TO CHANGE THE DEFAULT INTERATOMIC
DISTANCES USED IN DETERMINING BOND TYPES?N<
DO YOU DESIRE THAT ANY RINGS BE MADE RESONANT?Y<
TYPE YES OR NO AFTER EACH RING PROMPT
C4, C11, C10, C9, C8, C7, ?Y<
```

Figure 3-7. Interaction in %CONNECTMOL example (Figure 3-8 on reverse).

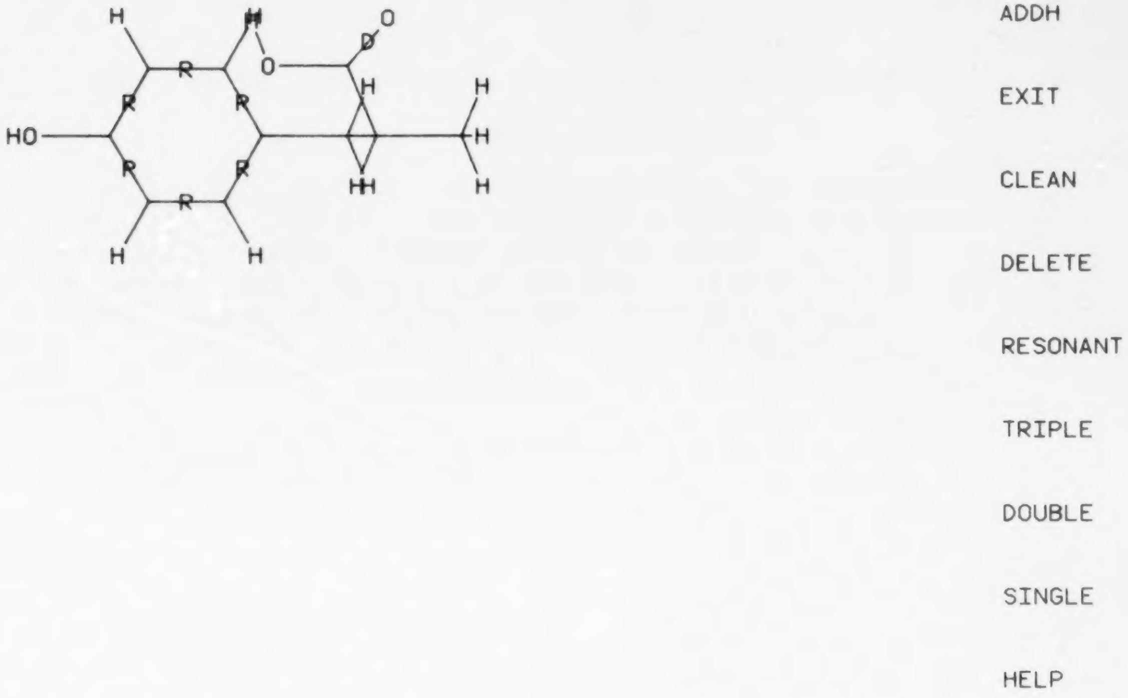

ADDH
EXIT
CLEAN
DELETE
RESONANT
TRIPLE
DOUBLE
SINGLE
HELP

Figure 3-9. %CONNECTMOL quiz.

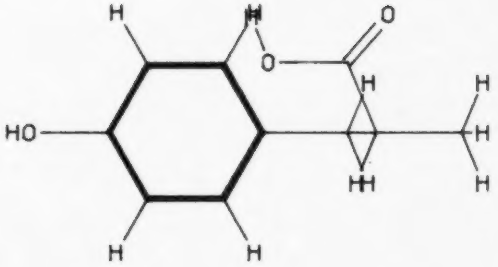


Figure 3-10. MOLQ displayed THREED COMPLETE after %CONNECTMOL.

Adding Hydrogen Atoms

To hydrogen balance a molecule, use the procedure

```
CALL %HBALANCE(M)
```

%HBALANCE <u>adds</u> hydrogen atoms to the molecule (M in this example) to satisfy the valences of all carbon and nitrogen atoms in the molecule. These added hydrogen atoms are referred to as "system-added hydrogens," and they are assigned neither sketch nor model coordinates by %HBALANCE. If necessary to correct a valence imbalance in the molecule, %HBALANCE also <u>removes</u> existing previously system-added hydrogens (defined as those hydrogens that do not have sketch coordinates). When you exit from the MAKE MOLECULE quiz in the normal course of making or editing a molecule, %HBALANCE is automatically called for you.

Adding Other Atoms

If you plan to add a new atom to an existing molecule that already has a three-dimensional model, you can use the function %GETNEWMVEC to determine the three-dimensional model coordinates that should be assigned to the new atom to make it lie in a position that you specify relative to other atoms in the molecule. It is called with a statement like the following:

```
NEWCOOR = %GETNEWMVEC(M,A1,A2,A3,DIST,BANGLE,TANGLE)
```

where you must specify the following arguments:

Argument	Type	Meaning
M	MOLECULE	Molecule to which new atom will be added
A1,A2,A3	FIXED	Indices of three atoms in M that already have model coordinates
DIST	FLOAT	Distance (in angstroms) from atom A1 to the new atom
BANGLE	FLOAT	Angle (in degrees) between a line (or bond) from atom A1 to atom A2 and a line (or bond) from atom A1 to the new atom

TANGLE	FLOAT	Dihedral angle (in degrees) between a line (or bond) from the new atom to atom A1 and a line (or bond) from atom A2 to atom A3 measured clockwise about a line (or bond) from atom A1 to atom A2

%GETNEWMVEC returns a 3-VECTOR (FLOAT ARRAY) that specifies the X, Y and Z coordinates of the new atom in model space. In the sample call above, the variable NEWCOOR would be set to the returned VECTOR.

%GETNEWMVEC Example

Suppose you wished to add a chlorine atom to the first carbon atom in a hydrocarbon chain in a molecule called M. For simplicity, let us assume that the first carbon in the chain has atom index 1 and each subsequent carbon atom in the chain is numbered sequentially. If you wished to specify that the bond length from the first carbon atom to the new chlorine atom is to be 1.7 angstroms, that the bond angle between the bond from the chlorine atom to the first carbon and the bond from the first carbon to the second carbon is to be 109.45 degrees, and that the chlorine is to be situated trans from the third carbon atom (i.e., with a dihedral angle of 180 degrees), you could use the following statement to add the new chlorine atom:

```
AI = %ADDNEWATOM(M,'CL', ,%GETNEWMVEC(M,1,2,3,1.7,109.45,180.0))
```

The system function %ADDNEWATOM is described in Part 2 of this chapter. In this example, AI is set to the FIXED index of the new atom.

Changing Dihedral Angles

To twist the model of part of a molecule about a single bond, use the procedure

```
CALL %TWISTMOL(M, A1, A2, ANG)
```

where A1 and A2 are the FIXED indices of two <u>non-ring</u> atoms connected by a single bond and ANG is the angle in degrees by which you wish to rotate the part of the molecule attached to atom A2. This angle is measured clockwise about the bond from atom A1 to atom A2.

Joining Molecules

%MERGEMOL adds one molecule to another, producing a molecule that preserves the relative sketch and model coordinates of the atoms of both original molecules in the new combined molecule. %MERGEMOL is called with a statement of the form:

```
CALL %MERGEMOL(M1,M2,SKTXFM,MDLXFM,A1,A2,BTYPE)
```

where the arguments have the following meanings:

Argument	Type	Meaning
M1	MOLECULE	Molecule that M2 is to be added to
M2	MOLECULE	Molecule to add to M1
SKTXFM	MATRIX	Transformer to apply to the sketch coordinates of M2 before adding them to M1
MDLXFM	MATRIX	Transformer to apply to the model coordinates of M2 before adding them to M1
A1	FIXED	Index of the atom in M1 that is to be bonded to an atom in M2
A2	FIXED	Index of the atom in M2 that is to be bonded to an atom in M1.
BTYPE	TEXT	Type of bond ('S', 'D', 'T') between atoms BATOM1 and BATOM2

See Chapter IV for an explanation of the transformers and matrices referred to in describing SKTXFM and MDLXFM above.

%MERGEMOL also preserves any special columns (such as the columns created by the MARK command for use in SELECTING options) that may exist in the ATOMS or BONDS tables of either of the original molecules. If a special column in one of the tables of M2 already exists in the corresponding table for M1, the values in that column from M2 are simply added to the preexisting column in the M1 table as the atoms or bonds from M2 are added. Otherwise, a new column is created with the same heading as the original column in the table of M2. Note that the public procedure EDITMODEL described in Chapter V can also be used to add one molecule to another.

%ADJUSTMOL Reminder

Since many of the procedures that we have described in this section modify the basic structure of a molecule, you should be careful to always include a call to %ADJUSTMOL after any series of calls to these procedures. The use of %ADJUSTMOL is described in Section 3A of this chapter.

Section 3E: Retrieving and Modifying Constraints

In Part 2 of this chapter, we described the gatekeeper procedures that allow you to retrieve and modify cells in the CONSTRAINTS table of a molecule referenced by constraint index. In this section, we describe procedures that allow reference to constraint values in a more convenient way -- not by their indices, but rather by the indices of the atoms and bonds that they affect. For each generic type of constraint that the PROPHET model builder recognizes there are two procedures: one that retrieves the value for that constraint (if it exists) for a particular atom or bond, and one that sets a value for that constraint for a particular atom or bond, replacing a previous constraint for the same atom or bond if it already exists. The use of constraints in computing molecular models is explained in detail in Part 2 of Chapter II.

The notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE	
CI	FIXED	Constraint index
B1,B2,B3	FIXED	Bond indices
AI	FIXED	Atom index
L	FLOAT	Bond length in angstroms
ANG	FLOAT	Angle in degrees
BARRAY	FIXED ARRAY	Array of bond indices
CCORCW	TEXT	Either 'CC' or 'CW'

Bond Length Constraints

To retrieve the length in angstroms that bond B1 in molecule M has been constrained to, use a statement like the following:

```
CI = %GETCONBONDLENGTH(M, B1, L)
```

Table 3-15: Procedures Used to Set and Retrieve Constraints

Calling Form	Usage
%GETCONBONDLENGTH(M,B1,L)	Returns (FIXED) index of bond length constraint for bond B1 in M.
CALL %SETCONBONDLENGTH(M,B1,L)	Sets or replaces bond length constraint for bond B1 in M.
%GETCONTORSION(M,B1,B2,B3,ANG)	Returns (FIXED) index of dihedral angle constraints about bond B2 in M.
CALL %SETCONTORSION(M,B1,B2, B3,ANG)	Sets or replaces dihedral angle constraint about bond B2 in M.
%GETCONDIRSANGLES(M,AI,CTYPE, BNDS,CVAL)	Returns (FIXED) index of directions or bond angles constraint about atom AI in M.
CALL %SETCONDIRECTIONS(M,AI BNDS,DIRS)	Sets or replaces directions constraint about atom AI in M.
CALL %SETCONBONDANGLES(M,AI, BNDS,ANGS)	Sets or replaces bond angles constraint about atom AI in M.
%GETCONCHIRAL(M,AI,B1,B2,B3, CCCW)	Returns (FIXED) index of chirality constraint about atom AI in M.
CALL %SETCONCHIRAL(M,AI,B1, B2,B3,CCCW)	Sets or replaces chirality constraint about atom AI in M.

Notation Conventions

M	MOLECULE	
AI	FIXED	Atom index
ANG	FLOAT	Angle in degrees
ANGS	FLOAT ARRAY	Bond angles in degrees
B1,B2,B3,B4	FIXED	Bond indices
BNDS	FIXED ARRAY	Bond indices
CCCW	TEXT	'CC' or 'CW'
CTYPE	TEXT	Constraint type, 'DIRECTIONS' or 'BOND ANGLES'
CVAL	VECTOR ARRAY or FLOAT ARRAY	Constraint value
DIRS	VECTOR ARRAY	Direction vectors
L	FLOAT	Length in angstroms

%GETCONBONDLENGTH sets the length that bond B1 is constrained to in the variable L and returns the FIXED index of the pertinent constraint, which CI is set to in this example. If there is no bond length constraint for bond B1, L is set to EMPTY and 0 is returned as the value of the function.

Conversely, to set a bond length constraint for a bond B1 in M, use the procedure

```
CALL %SETCONBONDLENGTH(M, B1, L)
```

If a bond length constraint already exists for bond B1, %SETCONBONDLENGTH replaces it with the new constraint you specified; otherwise the new constraint is added to the end of the CONSTRAINTS table. If you call %SETCONBONDLENGTH with L set to EMPTY, %SETCONBONDLENGTH removes any bond length constraint pertaining to bond B1 from the CONSTRAINTS table.

Dihedral Angle Constraints

To determine if there is a dihedral angle constraint specified about bond B2 in M, use the function

```
CI = %GETCONTORSION(M, B1, B2, B3, ANG)
```

In this example, %GETCONTORSION finds a constraint specified for bond B2. It then sets B1 and B3 to the indices of the two bonds adjacent to opposite ends of bond B2 that the constraint is specified for. It sets ANG to the dihedral angle constraint in degrees, between bonds B1 and B3, measured, as usual, clockwise about bond B2 looking from bond B1 toward bond B3. CI is set to the FIXED constraint index that %GETCONTORSION returns. If there is no dihedral angle constraint for bond B2 in M's CONSTRAINTS table, B1, B3 and ANG are set to EMPTY and %GETCONTORSION returns a function value of 0, which is assigned to CI in this example.

To set a dihedral angle constraint about bond B2 in M, use the procedure

```
CALL %SETCONTORSION(M, B1, B2, B3, ANG)
```

Before calling %SETCONTORSION, you must set B1 and B3 to the FIXED indices of two bonds adjacent to opposite ends of bond B2, and you must set ANG to the dihedral angle in degrees (measured in the way described in Chapter II, Part 2) you wish to set as the constraint. If you set B1, B2 or B3 to inappropriate bond indices, %SETCONTORSION simply returns after issuing an appropriate error message. Otherwise, it replaces any existing dihedral angle constraint about bond B2 with the new constraint you specify or, if necessary, adds a new constraint to the CONSTRAINTS table. If you call %SETCONTORSION with L set to EMPTY, the procedure deletes any dihedral angle constraints applying about bond B2 from the CONSTRAINTS table.

Bond Angle and Direction Constraints

In order to determine if there are any constraints in molecule M's CONSTRAINTS table specifying bond angle or bond direction constraints about atom AI, use a statement like

```
CI = %GETCONDIRSANGLES(M,AI,CTYPE,BARRAY,CVALUE)
```

CI is set to the FIXED constraint index returned by %GETCONDIRSANGLES; if there is no directions or bond angles constraint about atom AI, the function value returned is 0. If %GETCONDIRSANGLES succeeds in locating an appropriate constraint, it sets CTYPE to the TEXT type of the constraint (either 'BOND ANGLES' or 'DIRECTIONS'), BARRAY to the FIXED ARRAY of bond indices about atom AI to which the constraint applies, and CVALUE (which you should DECLARE as an ANY variable) to the value of the constraint found. If %GETCONDIRSANGLES finds a directions constraint, the value set into CVALUE is the VECTOR ARRAY of relative direction vectors for the bonds in BARRAY, while if it finds a bond angles constraint, the value set into CVALUE is the FLOAT ARRAY of angles applying between the bonds in BARRAY. The interpretation of these vectors or angles is described in Part 2 of Chapter II.

To set or replace bond direction constraints about atom AI in M, use the procedure

```
CALL %SETCONDIRECTIONS(M, AI, BARRAY, DIRS)
```

where DIRS is a VECTOR ARRAY of direction vectors, one for each bond in BARRAY. If the bond indices specified in BARRAY do not include all of the bonds about atom AI and no others, %SETCONDIRECTIONS simply returns after typing an appropriate error message. If there already is a bond directions or bond angles constraint that applies to atom AI, %SETCONDIRECTIONS replaces it with the new direction constraint; otherwise, %SETCONDIRECTIONS adds a new row to the CONSTRAINTS table. If you call %SETCONDIRECTIONS with DIRS set to EMPTY, the procedure deletes any bond angle or directions constraints applying to atom AI from the CONSTRAINTS table.

To set or replace bond angle constraints about atom AI in M, use the procedure

```
CALL %SETCONBONDANGLES(M, AI, BARRAY, ANGLES)
```

where ANGLES in a FLOAT ARRAY containing the angles in degrees between the bonds listed in BARRAY. As with %SETCONDIRECTIONS, if the bond indices specified in BARRAY do not include all of the bonds about atom AI and no others, %SETCONBONDANGLES simply returns after typing an appropriate error message.

As described in Part 2 of Chapter II, if atom AI is attached to two bonds, ANGLES must contain one angle; if atom AI is attached to three bonds, ANGLES must contain three angles; and if atom AI is attached to four bonds, ANGLES must contain six angles. If atom AI is attached to more than four bonds, you cannot specify bond angles constraints about it, but you can use directions constraints instead. If there already is a bond angles or directions constraint that applies to atom AI, %SETCONBONDANGLES replaces it with the new bond angles constraint; otherwise, %SETCONBONDANGLES adds a new row to the CONSTRAINTS table. If you call %SETCONBONDANGLES with ANGLES set to EMPTY, the procedure deletes any bond angles or directions constraint applying to atom AI from the CONSTRAINTS table.

Chirality Constraints

To determine if there is a chirality constraint specified for the bonds about atom AI in M, use a statement like

```
CI = %GETCONCHIRAL(M, AI, B1, B2, B3, CCORCW)
```

CI is set to the FIXED constraint index returned by %GETCONCHIRAL; if there is no chirality constraint about atom AI, the function value returned is 0. If %GETCONCHIRAL is successful in locating a chirality constraint for atom AI, it sets B1, B2 and B3 to the indices of the three bonds surrounding atom AI used to specify the constraint, and it sets CCORCW to the TEXT 'CW' or 'CC', depending on whether the sense of the chirality constraint is clockwise or counterclockwise as defined in Part 2 of Chapter II.

To set or replace a chirality constraint, use the procedure

```
CALL %SETCONCHIRAL(M, AI, B1, B2, B3, CCORCW)
```

where CCORCW is a TEXT containing either 'CC' or 'CW'. If bonds B1, B2 and B3 are not attached to atom AI, %SETCONCHIRAL simply returns after typing an appropriate error message. If there already is a chirality constraint that applies to atom AI, %SETCONCHIRAL replaces it with the new constraint; otherwise, it adds a new row to the CONSTRAINTS table. If you call %SETCONCHIRAL with CCORCW set to EMPTY, the procedure deletes any chirality constraint applying to atom AI from the CONSTRAINTS table. As explained in Part 2 of Chapter II, chirality constraints should be set only for atoms that are attached to exactly four bonds. Note that if there is already a bond angle or directions constraint for atom AI, it is inappropriate to set a chirality constraint for the same atom.

Section 3F: Substructure Search Tools

PROPHET contains a powerful mechanism for searching for instances of molecular fragments in other molecules; this *substructure search* mechanism is described in detail in Chapter II of this manual. Table 3-16 summarizes several procedures that allow you to use this mechanism.

Search Procedures

The function

```
FLAG = %CHEMCTNS(M1, M2)
```

returns the BOOL value TRUE if the molecule M2 is contained as a substructure of the molecule M1; TRUE is also returned if M1 and M2 are topologically identical. FALSE is returned when M2 is not contained in M1.

To determine exactly what atoms in molecule M1 correspond to the atoms in the fragment molecule M2, use the procedure

```
ATMARRAY = %SUBSEARCH(M1, M2, EXACT)
```

where EXACT is a BOOL that you have set to TRUE if M1 and M2 are to be tested to see if they are topologically identical or to FALSE if you only require that M2 be contained in M1. If %SUBSEARCH is successful in finding the structure of M2 in M1, it returns a FIXED ARRAY of atom indices in M1 that correspond to the atoms in M2, in the order that the atoms appear in M2's ATOMS table. This FIXED ARRAY is assigned to ATMARRAY in the example above. If the search is unsuccessful, %SUBSEARCH returns an EMPTY FIXED ARRAY as its functional value.

Screening Molecules

A *MOLSCREEN* is an encoding of the structure of a molecule for use by the substructure search procedures. It is more efficient than using the molecule itself. MOLSCREENs may be substituted for molecules as arguments to %CHEMCTNS and %SUBSEARCH (in fact, the first step in substructure search is to convert molecules into MOLSCREENs). The function

```
SCREEN(M)
```

returns a MOLSCREEN which describes the structure of the molecule M. This function may be used as the derivation text of a derived column of MOLSCREENs.

Table 3-16: Procedures Used in Substructure Search

Calling Form	Usage
%CHEMCTNS(M1,M2)	Returns (BOOL) TRUE if fragment molecule M2 is contained in M1, FALSE if it is not.
SCREEN(M)	Returns (MOLSCREEN) for M, for use in derived columns of tables.
%SUBSEARCH(M1,M2,EXACT)	Returns (FIXED ARRAY) of atom indices in M1 that correspond to atoms in M2; returns EMPTY if there is no match.

Notation Conventions

M	MOLECULE	
M1	MOLECULE or MOLSCREEN	
M2	MOLECULE or MOLSCREEN	Fragment being searched for
EXACT	BOOL	If TRUE, M1 and M2 are checked if they are identical. If FALSE, checks if M2 is contained in M1

Section 3G: Empirical Chemical Formulas

Every molecule in PROPHET has an empirical chemical formula associated with it. These formulas are constructed according to the rules used by the Chemical Abstracts Service, as described in Chapter II. Empirical formulas may also be handled separately from any molecules (as in an index of a number of chemical compounds) by using a data type known as <u>CHEMTEXT</u>. The PROPHET procedures for handling CHEMTEXT formulas are described in this section and summarized in Table 3-17.

The CHEMTEXT Data Type

In PL/PROPHET, CHEMTEXT is a complex data type consisting of exactly one subfield, the FORMULA, which is of type TEXT. As with other complex data types, if a variable called CT is declared to be CHEMTEXT, its single subfield can be referred to either as CT.FORMULA or as FORMULA OF CT.

Making and Checking Formulas

CCHEMTEXT is a function analogous to CTEXT or CFLOAT that converts an object to a valid CHEMTEXT formula, if that is possible. It may be used in a statement like the following:

```
CT = CCHEMTEXT(THING)
```

where CT is of type CHEMTEXT and THING may be of any type. If THING is a MOLECULE, CCHEMTEXT returns the molecule's FORMULA. Otherwise, CCHEMTEXT takes CTEXT(THING) and attempts to interpret it as a valid CHEMTEXT formula, according to the specifications described in Part 6 of Chapter II. CCHEMTEXT can correct certain types of notational errors in the formula represented by CTEXT(THING). For example, the statement

```
TYPE CCHEMTEXT('C8O3H14')
```

produces the following output:

```
C8 H14 O3
```

If the value given to CCHEMTEXT produces an ambiguous formula, CCHEMTEXT types a warning and makes a guess. For example,

```
TYPE CCHEMTEXT('COCL2')
```

Table 3-17: Procedures that Handle Empirical Chemical Formulas

Calling Form	Usage
CCHEMTEXT(THING)	Returns (CHEMTEXT) formula of a molecule or converts a text string into a CHEMTEXT formula if it can be done unambiguously.
%CONVCHEMTEXT(F)	Interactively creates and returns a proper (CHEMTEXT) formula based on F.
%CONVCHEMTEXTQT(F,AMBIG)	Silently creates and returns a proper (CHEMTEXT) formula based on F. If there was an ambiguity in F, AMBIG is set to TRUE.
%GETCHEMTEXTKEY(CT)	Returns (TEXT) sorting key for formula CT.

Notation Conventions

CT	CHEMTEXT	Empirical chemical formula
F	CHEMTEXT or TEXT	Proposed chemical formula
THING	MOLECULE or TEXT or CHEMTEXT	Object from which chemical formula is to be extracted

produces the output:

```
WARNING:  STRING COCL2 GIVEN TO CCHEMTEXT
PRODUCES AN AMBIGUOUS FORMULA
C CL2 O
```

If CCHEMTEXT cannot possibly convert the object you give it into a valid CHEMTEXT formula, it aborts with an appropriate error message.

The procedure %CONVCHEMTEXT engages you in an interactive dialog to help you convert a formula into a proper CHEMTEXT. It can be used in a statement like the following:

```
CT = %CONVCHEMTEXT(FORM)
```

where CT is a CHEMTEXT to which the functional value returned by %CONVCHEMTEXT is assigned. FORM must be either of type TEXT or of type CHEMTEXT. The following example illustrates the interactive nature of %CONVCHEMTEXT.

```
#TYPE %CONVCHEMTEXT('COCL2')$ THE CHEMTEXT
FORMULA "COCL2" IS AMBIGUOUS
PLEASE INDICATE WHICH ELEMENTS ARE PRESENT IN
THE STRING "COCL"
FIRST ELEMENT?CO$
NEXT ELEMENT(JUST GO TO END)?CL$
NEXT ELEMENT(JUST GO TO END)?$
THANK YOU!
CL2 CO
```

%CONVCHEMTEXT also has a quiet entry, which may be used in a statement like the following:

```
CT = %CONVCHEMTEXTQT(FORM, AMBIG)
```

AMBIG is a variable of type BOOL that %CONVCHEMTEXTQT sets to TRUE if it encounters any ambiguities in converting FORM which must be a TEXT or a CHEMTEXT. The following example illustrates the use of %CONVCHEMTEXTQT:

```
#DECLARE AMBIG BOOL$
#TYPE %CONVCHEMTEXTQT('COCL2', AMBIG)$
C CL2 O
#TYPE AMBIG$
 TRUE
```

Ordering Formulas

In Part 6 of Chapter II we described how tables can be sorted by columns of type CHEMTEXT and how comparisons between CHEMTEXT variables are carried out by the conventions used by the Chemical Abstracts Service. The method that PROPHET uses to make this ordering work out properly is to compute a TEXT key for each CHEMTEXT formula that sorts according to the proper conventions. You can use the function %GETCHEMTEXTKEY to retrieve the TEXT sorting key for any CHEMTEXT formula with a statement like the following:

```
CTKEY = %GETCHEMTEXTKEY(CT)
```

In this example, CT must be of type CHEMTEXT and the value returned is assigned to the TEXT variable CTKEY. Returning to the ACETONE molecule we have been using as an example throughout this manual, we could type the command:

```
TYPE %GETCHEMTEXTKEY(ACETONE.FORMULA)
```

to get the following key typed:

```
C 00003H 000060 00001
```

Nonstandard CHEMTEXT Formulas

At times you may wish to represent the formula of a molecule in a way that is not compatible with the standard CHEMTEXT conventions. For example, you may wish to represent the formula of ethyl ether (whose standard CHEMTEXT formula is $C_4H_{10}O$) as $(C_2H_5)_2O$. In general, PROPHET purposely makes it difficult for you to create non-standard CHEMTEXT formulas in order to preserve the integrity of the formula sorting conventions described in Part 6 of Chapter II. Clearly, it is not appropriate to store a non-standard formula in the FORMULA subfield of a molecule, since the FORMULA is automatically recalculated by the system each time %ADJUSTMOL is called. You may instead use the ATTRIBUTES subfield of a molecule to store a non-standard formula, which may even have the ATTRIBUTE label 'FORMULA'. The following commands would accomplish this for the ethyl ether molecule mentioned above:

```
DECLARE CT CHEMTEXT<GO>
CT.FORMULA = '(C2H5)2O'<GO>
CALL %SETATTRIBUTE(ETHYLETHER, 'FORMULA', CT)<GO>
```

Note that when you DISPLAY a CHEMTEXT formula in PROPHET, all of the numerical digits are lowered below the line. If you make non-standard formulas that include prefix numerals, they will also be lowered in displays.

Section 3H: Setting Up Selector Columns

In Chapter II we described the use of the MARK and UNMARK commands for selecting subsets of the atoms and bonds in a molecule for use in the commands

```
    DISPLAY molecule-name ...  SELECTING
```

and

```
    MAKE  MOLECULE   unused-name   FROM   molecule-name
SELECTING ...
```

The effect of the MARK ATOMS or MARK BONDS command is to set up a column in the ATOMS or BONDS table of a molecule with a heading determined by the subset name that you specified in the command. Such *selector columns* contain BOOL values; for each atom or bond that you MARK, the corresponding BOOL value is TRUE, for all other atoms or bonds the BOOL value is FALSE or EMPTY. The molecule programmer can freely establish *any* column in the ATOMS or BONDS table as a selector column subject to the following rules:

1. The column must be of type BOOL
2. The column index must *not* be in the range 1-7 for the ATOMS table, nor in the range 1-3 for the BONDS table.
3. The column heading must not begin with 'WIPKE'.

There are several approaches to setting up a selector column: one can add the column to the ATOMS or BONDS table using the ADD COLUMN command, or the %ADDCOLUMN or %INSERTCOLUMN procedures, and then set the cells using %SETCELL. Alternatively values could be supplied for the selector column by means of a derived column. For example, the statement:

```
    CALL %SETCDERIV(M.BONDS, 'RESRINGS',
    'CELL(''TYPE'') = ''R''');
```

would cause the selector column 'RESRINGS' to select the resonant bonds of molecule M. These selector columns can also be created and filled in with the procedures described in this section and summarized in Table 3-18.

Table 3-18: Procedures Used to Mark and Unmark Parts of Molecules

Calling Form	Usage
CALL %MARKATOMS(M,C,ATMS)	Causes those atoms of M whose indices are specified in ATMS to be marked in selector column C of M.ATOMS.
CALL %MARKBONDS(M,C,BNDS)	Causes those bonds of M whose indices are specified in BNDS to be marked in selector column C of M.BONDS.
CALL %UNMARKATOMS(M,C,ATMS)	Causes those atoms of M whose indices are specified in ATMS to be unmarked in selector column C of M.ATOMS.
CALL %UNMARKBONDS(M,C,BNDS)	Causes those bonds of M whose indices are specified in BNDS to be unmarked in selector column C of M.BONDS.

Notation Conventions

M	MOLECULE	
ATMS	FIXED ARRAY	Atom indices
BNDS	FIXED ARRAY	Bond indices
C	FIXED or TEXT	Column number or column heading

The notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
M	MOLECULE	
C	FIXED or TEXT	Column number or column heading
ATMS	FIXED ARRAY	Atom indices
BNDS	FIXED ARRAY	Bond indices

ATOMS Table Selector Columns

To mark the subset of atoms in molecule M whose indices are in ATMS, use the procedure

```
CALL %MARKATOMS(M, C, ATMS)
```

C is a TEXT column heading or a FIXED column number in the ATOMS table of M. If column C already exists and is of type BOOL, %MARKATOMS merely changes the entries in that column corresponding to the atom indices (row numbers) in ATMS to TRUE and leaves all other entries unchanged. If column C does not already exist, %MARKATOMS adds a selector column to the ATOMS table and sets the value for each atom listed in ATMS to TRUE.

To unmark atoms in a selector column in the ATOMS table, use the procedure

```
CALL %UNMARKATOMS(M, C, ATMS)
```

C must be a TEXT column heading or a FIXED column number for a valid selector column that already exists in the ATOMS table. %UNMARKATOMS sets the values in that column to FALSE for all atoms listed in ATMS and leaves all the other values unchanged.

BONDS Table Selector Columns

Similar procedures are available for manipulating BONDS table selector columns. To mark the subset of bonds in molecule M whose indices are in BNDS, use the procedure

```
CALL %MARKBONDS(M, C, BNDS)
```

To unmark bonds in an existing selector column in the BONDS table, use the procedure

```
CALL %UNMARKBONDS(M, C, BNDS)
```

Labeled Display Columns

In Chapter II we also described labeled molecule displays, in which additional labels are displayed with atoms and/or bonds. This feature is invoked using the following option appended to any molecule display command:

```
                 ATOMS
LABELING               [FROM] [COLUMN] col-id
                 BONDS
```

The meaning of this option is: in displaying atom or bond I of molecule M, append to any other display labels (e.g., element type, atom index) the TEXT

```
CTEXT(%GETCELL(M.ATOMS,I,col-id))
```

The specified column of the ATOMS or BONDS table of molecule M is referred to as a label column. Label columns can be set up by the user and referred to by the LABELING . . . option above. Label columns may be of any data type and have any heading. In general, label columns that you set up should be in columns beyond columns 1 to 7 of the ATOMS table and beyond columns 1 to 3 of the BONDS table. In some special circumstances, however, you may wish to use the information stored in one of the standard ATOMS or BONDS tables columns as labels. For example, to display our ACETONE molecule labeling the carbon atoms with C's as shown in Figure 3-11, use the command

```
DISPLAY ACETONE THREED COMPLETE NOTYPES
LABELING ATOMS BY COL 1
```

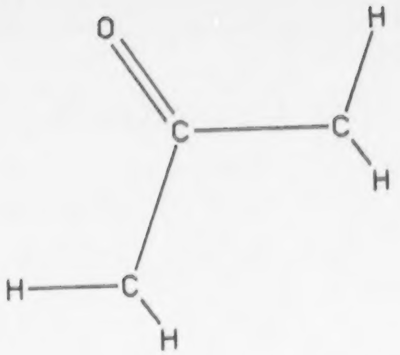


Figure 3-11. ACETONE displayed THREED COMPLETE, labeling atoms by column 1.

Label columns must be established by explicit table operations on the ATOMS and BONDS table of a molecule. There are no commands analogous to MARK, nor any procedures similar to %MARKATOMS.

Part 4: Multimolecules

A PROPHET MULTIMOLECULE represents a group of molecules with three-dimensional models and their spatial relationships. Part 4 of Chapter II describes several kinds of tools for working with multimolecules at top-level; for example, you can display a multimolecule and manipulate the display the same way as for single molecule displays. In this part, we present more detailed information about the MULTIMOLECULE data type and describe several system procedures that you may use to manipulate multimolecules.

Section 4A describes some of the subfields in the MULTIMOLECULE data type, while Section 4B documents the gatekeeper procedures that are available for PL/PROPHET programming involving multimolecules. The tools that are available for writing procedures that may be run by the INVOKE command in the MAKE/MANIPULATE MULTIMOLECULE quiz are described in Section 4C. Finally, Section 4D describes in some detail how the coordinate reference frames of the constituent molecules are related to the multimolecule's coordinate reference frame.

Section 4A: The Structure of Multimolecules

The MOLEC Subfield

Like a MOLECULE, a MULTIMOLECULE is a complex data type consisting of several subfields. Since a multimolecule is fundamentally a set of molecules, the MOLEC subfield is a MOLECULE ARRAY containing the constituent molecules. The molecules stored in the MOLEC subfield of a multimolecule are just like any other molecules, with one important exception. In a top-level molecule, the %TRANS subfield is a display transformer, as described in Part 1 of this chapter. In each molecule of the MOLEC subfield of a multimolecule, the %TRANS subfield plays a slightly different role. It is still a display transformer, but it is also involved in converting the model coordinates stored in the ATOMS table of the molecule from the molecule's own local coordinate reference frame to the coordinate reference frame of the entire multimolecule. For those who are interested in the relationships between the different coordinate reference frames in a multimolecule, Section 4D explains them in somewhat greater detail.

The %SYSCTR Subfield

The display transformer for the entire multimolecule is stored in the %SYSCTR subfield as a 4 x 3 FLOAT ARRAY ARRAY. See Chapter IV for a complete discussion of transformers.

The ATTRIBUTES Subfield

The ATTRIBUTES subfield of the multimolecule is an ANY ARRAY ARRAY in which arbitrary user-supplied information can be placed and accessed by name, in the same way as for the ATTRIBUTES of a molecule. These ATTRIBUTES may be set and accessed by the same gatekeeper procedures (%GETATTRIBUTE, %SETATTRIBUTE and %DELETEATTRIBUTE) used for setting and accessing the ATTRIBUTES of a molecule, as described in Part 2 of this chapter.

Section 4B: Multimolecule Gatekeeper Procedures

This section describes the procedural interfaces which allow you to access and manipulate multimolecules in contexts outside of the multimolecule quiz (see below). These gatekeepers are summarized in Table 3-19. The notation conventions for the variable names used in the sample procedure calls in this section are as follows:

Variable Name	Data Type	
MM	MULTIMOLECULE	
M	MOLECULE	
IM	FIXED	Molecule index
IA	FIXED	Atom index
N	FIXED	Number of molecules in a multimolecule
VMM	VECTOR	Model coordinates of an atom in multimolecule coordinate reference frame
XMM	MATRIX	Transformer (see Chapter IV)

Making and Copying Multi-molecules

The procedure

```
CALL %MAKEMULTIMOL(MM)
```

causes the structure of the multimolecule MM to be initialized so that component molecules can be added; MM must have been previously declared to be a MULTIMOLECULE.

To make a new multimolecule that is an exact copy of an existing multimolecule, use the function

```
NEWMM = %MULTIMOLCOPY(MM, MMTYPE)
```

Set the TEXT variable MMTYPE to 'DISK' if you wish all

Table 3-19: Multimolecule Gatekeeper Procedures

Calling Form	Usage
CALL %MAKEMULTIMOL(MM)	Initializes structure of MM, which must be previously declared to be a multimolecule.
%MULTIMOLCOPY(MM,DISKCORE)	Returns (MULTIMOLECULE) of specified type that is a copy of MM.
%MMLASTMOL(MM)	Returns (FIXED) index of last molecule in multimolecule MM.
%MMGETMOL(MM,IM)	Returns (MOLECULE), the IMth molecule in multimolecule MM.
CALL %MMADDMOL(MM,M,IM)	Adds molecule M to multimolecule MM, sets IM to index of new molecule.
CALL %MMDELETEMOL(MM,IM)	Deletes molecule IM of multimolecule MM.
%MMGETMVEC(MM,IM,IA)	Returns (VECTOR) coordinates of atom IA in molecule IM of multimolecule MM, in multimolecule reference frame.
CALL %MMSETMVEC(MM,IM,IA,VMM)	Sets multimolecule reference frame coordinates of atom IA of molecule IM in MM to VMM.
CALL %MMMANIPMOL(MM,IM,XMM)	Applies transformation XMM (in reference frame of multimol) to molecule IM of MM.
CALL %ADJUSTMULTIMOL(MM)	Updates system subfields of multimolecule MM after manipulation.
CALL %CONVERT2DISKMM(MM)	Converts all molecules in MM to disk-based molecules.

Table 3-19: Multimolecule Gatekeeper Procedures (cont'd)

Calling Form	Usage
CALL %CONVERT2COREMM(MM)	Converts all molecules in MM to core-based molecules.
%TESTDISKMM(MM)	Returns (BOOL) TRUE if any molecule in MM is disk-based.

Notation Conventions

MM	MULTIMOLECULE	
M	MOLECULE	Component of multimolecule
IM	FIXED	Index of molecule in multimolecule
IA	FIXED	Atom index
VMM	VECTOR	Atom coordinates in multimolecule reference frame
XMM	MATRIX	Transformer in multimolecule reference frame
DISKCORE	TEXT	'DISK', 'CORE' or EMPTY

the component molecules of the new multimolecule (NEWMM in this example) to be made disk-based, or set it to 'CORE' if you wish all the component molecules to be core molecules. If you leave the second argument empty, as in the example

```
NEWMM = %MULTIMOLCOPY(MM, )
```

then all of the component molecules of MM will remain the same type (disk or core) in NEWMM.

Finding the Molecules in a Multimolecule

The function

```
N = %MMLASTMOL(MM)
```

returns a FIXED value (assigned to the variable N in this example) which is the number of component molecules in the multimolecule MM. For the gatekeepers below which take molecule arguments, each component molecule is referenced by its <u>index</u>, and only indices in the range 1 to %MMLASTMOL(MM) are valid.

The function

```
M = %MMGETMOL(MM,IM)
```

returns the IMth component molecule in the multimolecule MM. Once the molecule has been retrieved, it can be accessed and modified as if it were a top-level molecule.

WARNING!

If you alter the molecule after retreiving it, it is necessary to call %ADJUSTMOL, just as if it were a top-level molecule. In addition, since altering the molecule alters the multimolecule of which it is a component, it is necessary to call %ADJUSTMULTIMOL (see below) as well.

Adding and Deleting Molecules

The procedure

```
CALL %MMADDMOL(MM,M,IM)
```

causes the molecule M to be added to the multimolecule MM, and sets the FIXED variable IM to the index of M in MM.

WARNING!

> %MMADDMOL does not make a copy of molecule M before adding it to the multimolecule. Thus, any manipulations performed on the IMth component molecule will be reflected in the top-level copy of M, and vice-versa. For this reason, if the top-level copy of M is to be used after adding it to the multimolecule, it is better to pass %MOLCOPY(M,) as the argument to %MMADDMOL; this is particularly important in building up a multimolecule out of several instances of the same molecule.

The IMth molecule of multimolecule MM can be deleted with the procedure

```
CALL %MMDELETEMOL(MM,IM)
```

Model Coordinates

Use the function

```
VMM = %MMGETMVEC(MM,IM,IA)
```

to retrieve the 3D coordinates of the atom with index IA of the molecule with the index IM in multimolecule MM, expressed in the multimolecule reference frame.

In this example, the coordinates are assigned to the VECTOR VMM. To alter the position of the IAth atom of the IMth molecule of multimolecule MM, use the procedure

```
CALL %MMSETMVEC(MM,IM,IA,VMM)
```

where VMM is a VECTOR containing the new position of the atom in the multimolecule reference frame.

WARNING!

> If you use %MMSETMVEC to alter the structure of a component molecule, you must call %ADJUSTMOL for the affected molecule, as well as %ADJUSTMULTIMOL for the multimolecule as a whole.

Moving a Molecule in a Multi-molecule

Often, one wishes to alter the structure of a multimolecule by moving one of the component molecules with respect to the multimolecule reference frame. The procedure

```
CALL %MMMANIPMOL(MM,IM,XMM)
```

moves the IMth molecule of multimolecule MM by applying the transformer (MATRIX) XMM to the coordinates of its atoms in the multimolecule reference frame (transformers are described in Chapter IV of this manual). For example, to turn the IMth molecule of multimolecule MM by 30 degrees around an axis defined by two of its atoms, one could write the following procedure:

```
/* MMTURNMOL turns the IMth molecule of the
   multimolecule MM 30 degrees around the axis
   defined by its Ith and Jth atoms.   */

   MMTURNMOL:  PROCEDURE (MM,IM,I,J);

   DECLARE                     /* Arguments */
    MM MULTIMOLECULE,
    IM FIXED, /* Molecule index */
    I FIXED,  /* Atom index */
    J FIXED;  /* Atom index */

  DECLARE                      /* Locals */
    VMI VECTOR,                /* MM coords of atom I */
    VMJ VECTOR,                /* MM coords of atom J */
    ROTOR MATRIX;              /* Rotation transformer */

    /* Extract multimolecule ref. frame coordinates */

     VMI = %MMGETMVEC(MM,IM,I);
     VMJ = %MMGETMVEC(MM,IM,J);

    /* Generate rotation transformer by specifying
       center, axis direction, and angle */

     ROTOR = %ROTATION(VMI,VMJ-VMI,30.);

    /* Apply transformer to molecule */

     CALL %MMMANIPMOL(MM,IM,ROTOR);
     CALL %ADJUSTMULTIMOL(MM); /* Clean up */
    END;                       /* MMTURNMOL */
```

Notice that the rotation is generated using the multimolecule reference frame coordinates (from %MMGETMVEC). The transformer XMM must be expressed in the multimolecule coordinate frame. Notice also that when %MMMANIPMOL is called, it alters the structure of the multimolecule, so %ADJUSTMULTIMOL must be called as well.

The Attribute Gatekeepers

The ATTRIBUTE gatekeepers %GETATTRIBUTE, %SETATTRIBUTE and %DELETEATTRIBUTE, described in Part 2 of this chapter for molecules, work identically for multimolecules.

Adjusting a Multimolecule

The procedure

```
CALL %ADJUSTMULTIMOL(MM);
```

should be called after using any routines which might affect the structure of the multimolecule; like %ADJUSTMOL, it brings several subfields of the multimolecule into a consistent state for display.

Disk-Based Multimolecules

A disk-based multimolecule is a multimolecule for which at least one of its constituent molecules is disk-based, which is to say that at least one constituent molecule has ATOMS, BONDS (and CONSTRAINTS) tables that are disk-resident. Procedures analogous to those for converting molecules between their disk-based and core-based forms exist for multimolecules as well. The procedure

```
CALL %CONVERT2DISKMM(MM)
```

will cause all of the tables associated with <u>all</u> of multimolecule MM's constituent molecules to be made into disk-based tables. Conversely, the procedure

```
CALL %CONVERT2COREMM(MM)
```

will cause all of the tables associated with <u>all</u> of multimolecule MM's constituent molecules to be made into temporary core-based tables. In both cases, any constituent molecules that are already of the desired type will be left untouched.

best copy available

To determine if a multimolecule is disk-based, use the function procedure

```
FLAG = %TESTDISKMM(MM)
```

which returns a BOOL value of TRUE (which is assigned to the variable FLAG in this example) if any of the component molecules of multimolecule MM are disk-based.

Section 4C: The Multimolecule Quiz and INVOKE

The PROPHET commands

```
MAKE MULTIMOLECULE unused-name
```

and

```
MANIPULATE multimolecule-name [NOBACKUP]
```

cause execution of the multimolecule quiz, an interactive display environment in which the multimolecule can be manipulated and its structure altered (see Chapter II). The set of tools provided by the quiz is reasonably flexible, but there are clear limitations to what the multimolecule quiz can do. For this reason, the INVOKE command has been added to the quiz. INVOKE allows you to call your own external procedures to interact with the multimolecule being made or manipulated.

When you indicate the INVOKE light-button, the quiz prompts you to enter the name of a procedure. Suppose, for example, you type UPROC. After checking that UPROC is present in the current workspace as a valid, defined PROCEDURE, the quiz, in effect, executes the following statements:

```
CMDTOK = 'UPROC';
CALL UPROC(CMDTOK);
```

where CMDTOK is an ANY variable. For this reason, any invokable procedure must take a single ANY argument. Upon return from UPROC, the value of CMDTOK is taken to be the next command. Thus, if your procedure sets the argument to 'EXIT' before returning, the EXIT command will be executed next. Only a single command can be "stacked" in this fashion. If no command is to be stacked, UPROC should CLEAR its argument before returning to the main body of the quiz.

In order for the INVOKE command to be useful, it must be possible for the user-supplied procedures to interact with the multimolecule being made or manipulated. To do this, your procedure can call any of the multimolecule gatekeepers described above, as well as several procedures which have meaning only within the context of the multimolecule quiz (see Table 3-20). Examples of procedures that can be used with the INVOKE light button are given in Part 5 of this chapter.

Table 3-20: Multimolecule Quiz Gatekeepers

Calling Form	Usage
%MMQMULTIMOL	Returns (MULTIMOLECULE) which is the one being made or manipulated.
CALL %MMQPOINTAT(TYP,IM,IAORB)	Allows detection of user tablet indication.
CALL %MMQREFRESH	Must be called to update display after any manipulations.

Notation Conventions

TYP	TEXT	'ATOM', 'BOND', 'MOLECULE'
IM	FIXED	Molecule index
IAORB	FIXED	Atom or bond index

Table 3-21: Multimolecule Transformer Gatekeepers

Calling Form	Usage
%MMGETMXFM(MM,IM)	Returns (MATRIX) transformer mapping coordinate frame of molecule IM in MM into multimolecule reference frame.
CALL %MMSETMXFM(MM,IM,X)	Sets transformation associated with molecule IM in MM to X

Notation Conventions

MM	MULTIMOLECULE	
IM	FIXED	Molecule index
X	MATRIX	Transformer

WARNING!

Any attempt to use the intra-quiz routines described below (i.e., those whose names begin with %MMQ) outside the context of the quiz, or to interact with the multimolecule in other than the prescribed fashion, is likely to damage your multimolecule.

Identifying a Multimolecule

The function

```
MM = %MMQMULTIMOL
```

returns the multimolecule which is being manipulated by the multimolecule quiz. It is not a variable, but rather a function with no arguments, and thus cannot be used on the left side of an assignment statement.

Allowing a Tablet Indication

The procedure %MMQPOINTAT can be called to allow a program to receive a tablet indication from the quiz user. The statement

```
CALL %MMQPOINTAT(TYP,IM,IAORB)
```

causes the program to wait for you to point to something on the data tablet. When you do, the procedure returns with its arguments set to reflect the point indicated: the TEXT variable TYP is set to 'ATOM' or 'BOND' if an atom or bond was pointed to, to 'MOLECULE' if some point within a molecule was indicated, or is set to EMPTY otherwise. Except for the last case, the <u>index</u> of the pointed-to molecule is assigned to the FIXED variable IM. If an atom or bond was indicated, its index is assigned to the FIXED variable IAORB; otherwise, IAORB is set to EMPTY.

Cleaning up After INVOKE

The procedure %MMQREFRESH must be called within the INVOKE environment whenever the structure of the multimolecule or any of its components is altered, e.g., by %MMMANIPMOL. It erases the screen and redisplays the multimolecule in its altered state. Failure to call %MMQREFRESH will, at best, vitiate the effects of the alterations, and, at worst, damage the 3D structure of the multimolecule. It is good practice to end any invokeable procedure which alters the multimolecule with the statements:

```
CALL %ADJUSTMULTIMOL(MM);
CALL %MMQREFRESH;
```

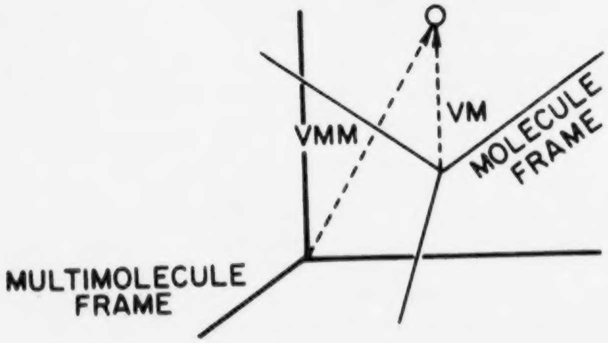
Section 4D: Implementation of Multimolecules

This section describes details of the implementation of coordinate frames for multimolecules. It may be skipped by the casual reader, since we have already presented all the information needed to program with multimolecules. This section assumes familiarity with the PL/PROPHET facilities for handling vectors, matrices, and transformers, as described in Chapter IV of this manual.

Reference Frames

The coordinates stored in columns 5-7 of the ATOMS table in a molecule define the 3D position of atoms in a coordinate reference frame local to the molecule. These same coordinates are returned as the result of the molecule gatekeeper %GETMVEC. In the context of multimolecules, it is necessary to transform these atom coordinates into the multimolecule coordinate reference frame before performing any geometric computation (e.g., computing interatomic distances between atoms in different component molecules, generating transformers for %MMMANIPMOL, etc.). For this purpose, the multimolecule gatekeeper %MMGETMVEC returns coordinates of atoms in the multimolecule reference frame.

One can describe the relation between the reference frame of a multimolecule MM and the frame of one of its component molecules M by a transformation X: X, applied to a point with coordinates VM in the molecule frame, yields VMM, the coordinates of that point in the multimolecule frame, thus:



The transformation X is a combination of rotation and translation, with no scale change.

PL/PROPHET allows one to describe the transformation X by a transformer (MATRIX), so that one may write:

```
VMM = VM XFM X;
```

The following two gatekeepers, summarized in Table 3-21, provide access to the transformation (transformer) associated with the molecules of a multimolecule:

Reference Transformers

The function

```
X = %MMGETMXFM(MM,IM)
```

returns the transformer which maps the coordinate frame of the IMth molecule of multimolecule MM into the reference frame of MM.The procedure

```
CALL %MMSETMXFM(MM,IM,X)
```

sets the transformation for the IMth molecule of MM to the transformer X; X is checked to insure it is orthonormal (pure rotation and translation) and aborts if it is not.

With these two gatekeepers, one can derive several others already described. For example, we could write the gatekeeper %MMGETMVEC as follows:

```
/*  %MMGETMVEC returns the (VECTOR) coordinates of atom
IA in molecule IM of multimolecule MM in multimolecule
reference frame. */

   %MMGETMVEC:  PROCEDURE(MM,IM,IA);
    RETURNS VECTOR;

   DECLARE                        /* Arguments */
    MM  MULTIMOLECULE, /* [IN] */
    IM  FIXED,                    /* Mol. index [IN] */
    IA  FIXED;                    /* Atom index [IN] */

   DECLARE                        /* Locals */
    VM  VECTOR,                   /* Coords in mol. */
    X   MATRIX,                   /* Transformer to MM */
    M   MOLECULE,
    VMM VECTOR;                   /* Coords in multimol. */

    M = %MMGETMOL(MM,IM);   /* Retrieve mol. */
    X = %MMGETMXFM(MM,IM); /* and transformer */
    VM  = %GETMVEC(M,IA);  /* Coords in mol. */
    VMM = VM XFM X;         /* and in multimol */
    RETURN(VMM);

   END;       /* MMGETMVEC */
```

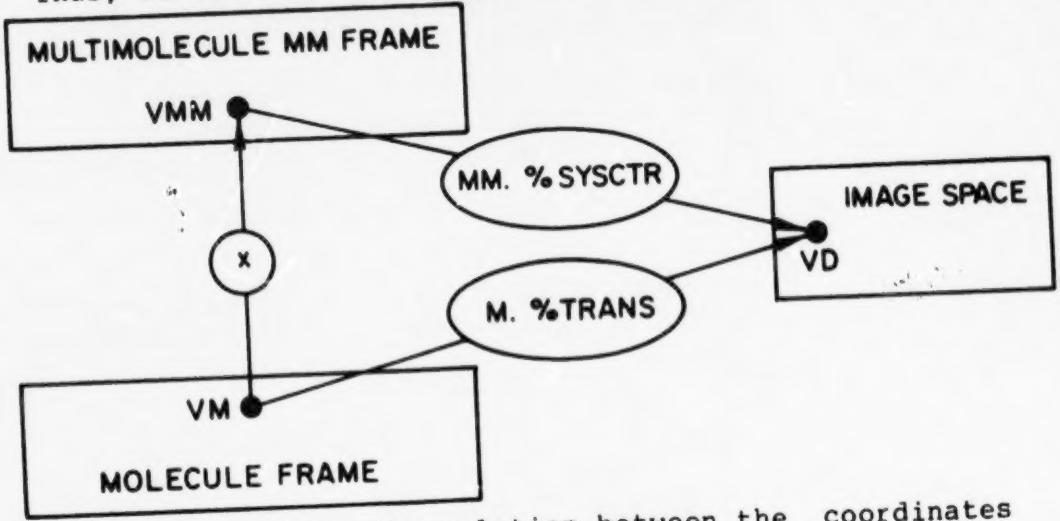
Similarly, the gatekeeper %MMMANIPMOL could be written as follows:

```
/*  %MMMANIPMOL applies transformation XMM (in
reference frame of multimol) to molecule IM of MM. */

   %MMMANIPMOL:             PROCEDURE(MM,IM,XMM);

DECLARE                          /* Arguments */
    MM  MULTIMOLECULE, /* [IN] */
    IM  FIXED,                   /* Mol. index [IN] */
    XMM MATRIX;                  /* Transformer [IN] */

DECLARE                          /* Locals */
    X   MATRIX,                  /* Old transf. */
    NX  MATRIX;                  /* New transf. */

    X = %MMGETMXFM(MM,IM); /* Get old transf. */
    NX = X XFM XMM;         /* Compute new one */
    CALL %MMSETMXFM(MM,IM,XMM); /* Transform IM */

 END;          /* %MMMANIPMOL */
```

Transformer Interrelationships

The transformation X is associated with the molecule M of a multimolecule in a somewhat indirect way: all molecules (whether they are components of multimolecules or not) have a subfield called %TRANS, which is a transformer (MATRIX) which maps stored coordinates in the molecule frame into another 3D reference frame which corresponds to the the displayed image; call the latter frame image space. When a molecule is displayed, the %TRANS subfield is used to map molecule coordinates into image space coordinates; altering the display (e.g., by using TURN or FREEZE) alters the %TRANS subfield associated with the molecule.

In the multimolecule itself, there is a subfield called %SYSCTR which performs an analogous function to the %TRANS subfield of molecules: it maps coordinates in the multimolecule reference frame into image space. Thus, we have the following situation:



The diagram shows the relation between the coordinates of a point in the multimolecule reference frame (VMM), its coordinates in the frame of molecule M (VM), and the coordinates in the image space (VD):

```
VD = VMM XFM MM.%SYSCTR
VD = VM XFM M.%TRANS
```

By substituting one equation into the other, we get:

```
VMM = VM XFM M.%TRANS XFM %INVXFM(MM.%SYSCTR)
```

But we have already defined X to be the transformation which maps VM into VMM:

```
VMM = VM XFM X
```

so we conclude:

```
X = M.%TRANS XFM %INVXFM(MM.%SYSCTR)
```

Given this last result, then, we can write the code for %MMGETMXFM, as follows:

```
/* %MMGETMXFM obtains the transformer that converts
   the coordinates of the atoms in molecule indexed
   by IM in multimolecule MM from the local coordinate
   system in the molecule to the multimolecule
   frame of reference    */

   %MMGETMXFM: PROCEDURE (MM,IM);
    RETURNS MATRIX;

   DECLARE                     /* Arguments */
    MM        MULTIMOLECULE, /* [IN] */
    IM        FIXED;         /* Molecule index [IN] */

   DECLARE                     /* Locals */
    M         MOLECULE;

    M = %MMGETMOL(MM,IM);    /* Get molecule */
    RETURN (M.%TRANS  XFM  %INVXFM(MM.%SYSCTR));

   END;        /* %MMGETMXFM */
```

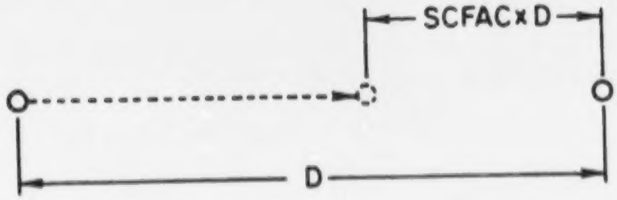
Part 5: Examples of Molecule Programming

This section presents several sample programs which use some of the molecule procedures described earlier in this chapter.

```
/* MOLCTR takes a molecule with a model as argument,  and
   returns  a  VECTOR  which  is its center of mass.  The
   algorithm uses the fact that the center of mass is a
   vector which is the weighted average of the coordinate
   vectors of the atoms, where the weighting is by  atomic
   mass */

   MOLCTR:  PROCEDURE (MOLQ);
        RETURNS VECTOR;

   DECLARE                 /* Procedure arguments */
       MOLQ    MOLECULE;   /* Mole. with model [IN] */

         DECLARE          /* Local variables */
       IATOM   FIXED,              /* Atom index */
       ATYPE   TEXT,               /* Atom type */
       ATMWT   FLOAT,              /* Atomic weight */
       SUMWT   FLOAT,              /* Sum of weights */
       ATMVC   VECTOR,             /* Atom location */
       SUMWVC  VECTOR;             /* Weighted vector sum */

 SUMWT =0.0;                     /* Initialize */
 SUMWVC = [0., 0., 0.];          /* Initialize */

 IF (NOT(%GETFLAG(MOLQ,%M%MDL))) THEN
   TYPE 'NO MODEL FOR MOLECULE';

 ELSE DO IATOM = 1 TO %LASTATOM(MOLQ);
     ATYPE = %GETATOMTYPE(MOLQ,IATOM);
     CALL %FINDELEMENT(ATYPE,,ATMWT,,);
     ATMVC = %GETMVEC(MOLQ,IATOM);

            IF (NOTEMPTY(ATMWT) AND
              NOTEMPTY(ATMVC[1])) THEN DO;
                SUMWT = SUMWT +ATMWT;
         SUMWVC = SUMWVC + ATMWT*ATMVC;
       END;                /* THEN DO */
 END;                      /* DO IATOM loop*/

 IF (SUMWT > 0.0) THEN
   RETURN ((1./SUMWT)*SUMWVC);

    ELSE RETURN (SUMWVC);        /* which = [0 0 0] */

   END;  /* MOLCTR */
```

```
/* COUNTCH3 takes a molecule as  argument  and  returns  a
   count  of  the  number  of  -CH3  groups (returns 1 for
   Methane), by counting the number of Carbons which are
   joined by single bonds to 3 or more hydrogens */

   COUNTCH3:  PROCEDURE (MOLQ);
       RETURNS FIXED;

   DECLARE                           /* Arguments */
          MOLQ    MOLECULE;          /* [IN] */

   DECLARE                           /* Locals */
 IATOM   FIXED,               /* Atom index */
 COATMS  FIXED ARRAY,         /* Its neighbors */
 COBNDS  FIXED ARRAY,         /* Its bonds */
 ICO     FIXED,               /* Neighbor index */
 NH      FIXED,               /* NO. of -H's */
 NCH3    FIXED;               /* CH3 count */

 NCH3 = 0;                    /*Initialize */

 DO IATOM = 1 TO %LASTATOM(MOLQ);
          IF (%GETATOMTYPE(MOLQ,IATOM) >< 'C')
            THEN DONEXT;
          COATMS = %ATOMNEIGHBORS(MOLQ,IATOM);
          COBNDS = %GETBONDA(MOLQ,IATOM);

        /* NH counts the number of H's singly
           bonded to this Carbon */

   NH = 0;

   DO ICO = 1 TO TOP(COATMS);
     IF (%GETBONDTYPE(MOLQ,COBNDS[ICO]) = 'S'
         AND
          %GETATOMTYPE(MOLQ,COATMS[ICO]) = 'H')
               THEN NH = NH + 1;
   END;                               /* DO ICO loop */
   IF (NH >= 3) THEN NCH3 = NCH3 + 1;
  END;                                /* DO IATOM loop */

  RETURN (NCH3);

     END;                             /* COUNTCH3 */
```

```
/* MMQSCALEDIST is a procedure to be INVOKEd within the
   multimolecule quiz. It asks the user to point to two
   atoms in different molecules, then to type a factor by
   which to scale their distances; the first molecule is
   then translated along the line between the atoms,  to
   effect the change of distance, thus:
```



```
 */

   MMQSCALEDIST:  PROCEDURE(CMDTOK);

   DECLARE                            /* Arguments */
 CMDTOK  ANY;               /* Std. for INVOKE [IN] */
   DECLARE                            /* Locals */
 MM      MULTIMOLECULE,
 TYP     TEXT,                        /* From %MMQPOINTAT */
 IM1     FIXED,                       /* First mol. */
 IA1     FIXED,                       /* First atom */
 IM2     FIXED,                       /* Second mol. */
 IA2     FIXED,                       /* Second atom */
 VMM1    VECTOR,                      /* First position */
 VMM2    VECTOR,                      /* Second position */
 SCFAC   FLOAT,                       /* Scale factor */
 TV      VECTOR;                      /* Translation */

 DO WHILE (EMPTY(IM1));
         TYPE 'POINT TO AN ATOM';
         CALL %MMQPOINTAT(TYP,IM1,IA1);
         IF (TYP='ATOM') THEN DOEXIT;
                 ELSE CLEAR IM1;
 END;            /* DO WHILE FOR IM1 */

 DO WHILE (EMPTY(IM2));
         TYPE 'POINT TO ATOM IN DIFFERENT MOL.';
         CALL %MMQPOINTAT(TYP,IM2,IA2);
```

```
        IF (TYP= 'ATOM') THEN
                IF (IM1 >< IM2) THEN DOEXIT;
        CLEAR IM2;
END;            /* DO WHILE for IM2 */

MM = %MMQMULTIMOL;
VMM1 = %MMGETMVEC(MM,IM1,IA1);
VMM2 = %MMGETMVEC(MM,IM2,IA2);

SCFAC = %GETFLOAT('ENTER DIST. SCALE FAC.',FALSE);

  /* Compute the vector by which molecule 1 must
     be translated to scale distance */

         TV = (1.0 -SCFAC) * (VMM2 - VMM1);

  /* Apply translation transformer, then clean up */

         CALL %MMMANIPMOL (MM,IM1,%TRANSLATEV(TV));
         CALL %ADJUSTMULTIMOL(MM);
          CALL %MMQREFRESH;
         CLEAR CMDTOK;         /* Don't stack any command */

  END;                   /* MMQSCALEDIST */
```

4. Vectors and Matrices

4. Vectors and Matrices

Chapter IV.

VECTOR, MATRIX AND TRANSFORMATION PROCEDURES AND FUNCTIONS

One of the distinctive features of the PL/PROPHET programming language is that it enables you to manipulate high-level data types such as TABLEs and MOLECULEs, as well as simpler data types containing numbers and character strings. A good example of this feature is the PL/PROPHET facility for handling vectors and matrices. The vector and matrix facilities were originally designed to support the geometric computations inherent in describing and displaying MOLECULEs, but they have a broad range of other applications as well.

The VECTOR and MATRIX data types and operations have two outstanding advantages:

1. Readability: manipulations involving vectors and matrices are written in a short natural notation.

2. Efficiency: the vector and matrix operators are considerably faster and more storage-efficient than equivalent routines written in PL/PROPHET without the use of these operators.

This chapter consists of five parts: Part 1 describes the representation of vectors and matrices as PL/PROPHET data types and introduces some notation conventions used in the remainder of this chapter. Parts 2 and 3 describe the operators and procedures used to manipulate vectors and matrices. Part 4 introduces the concept of the transformer, a matrix used for three-dimensional transformations, and describes the operators and procedures that may be used with transformers. Finally, Part 5 provides some examples of PL/PROPHET procedures that use vectors, matrices and transformers.

Part 1: Representation of Vectors and Matrices

For purposes of this discussion, *vectors* are defined to be one-dimensional arrays of real numbers, and matrices are defined to be M-row by N-column rectangular arrays of real numbers. These two data types are represented in PL/PROPHET by the data types FLOAT ARRAY and FLOAT ARRAY ARRAY respectively. In fact, the system allows the interchangable use of the equivalent data types, i.e.,

```
DECLARE V VECTOR;
DECLARE M MATRIX;
```

is equivalent to:

```
DECLARE V FLOAT ARRAY;
DECLARE M FLOAT ARRAY ARRAY;
```

or to:

```
DECLARE V(*) FLOAT;
DECLARE M(*,*) FLOAT;
```

Vectors and matrices may be used as top-level variables, as local variables and as procedure arguments. In short, they may be used in the same ways as any other data type in PL/PROPHET. They may be constructed and their components may be accessed by the available array-handling mechanisms, such as:

```
V = [1., 0., 0.];   /* ARRAY CONSTANT */

M(1,1) = 1.0;  M(1,2) = 0.0;  /* SUBSCRIPTING */
M(2,1) = 0.0;  M(2,2) = V(1);
```

In the latter case, the first dimension (or subscript) of a matrix corresponds to a row number, while the second corresponds to a column number.

Pitfalls

There are several mistakes to avoid when working with vectors and matrices in PL/PROPHET:

1. VECTOR is the equivalent of (*) FLOAT, *not* FLOAT(3). Thus, when we speak of an N-vector, it is actually a "variable length array whose TOP is N." The same is true of a MATRIX.

2. A scalar (FLOAT) is not the same thing as a one-vector, which is not the same thing as a one by one matrix, even though all three might be TYPEd out identically by PROPHET.

3. The temptation to build up vectors and matrices using array constants must be resisted if any elements are variable; e.g.

```
X = 13.;  Y = 3 * X**2;  Z = Y - 2.*X;

V = [X, Y, Z];
```

   will not have the desired effect in a PL/PROPHET procedure. Instead, one must say:

```
X = 13.;  Y = 3 * X**2;  Z = Y - 2*X;
V(1) = X;  V(2) = Y;  V(3) = Z;
```

Notation Conventions

The descriptions below use a mixture of mathematical and computer notation:

element:	One of the numbers in a matrix or vector
N-vector:	A vector with N elements
MxN Matrix:	A matrix with M rows and N columns
size:	The number of elements in a vector The number of rows, columns in a matrix
Vi:	The i-th element of vector V
Mi,j:	Element of matrix M in row i, column j

Part 2: Algebraic Operators

As Table 4-1 shows, many of the operations on vectors and matrices are provided in the form of arithmetic operators. Some of these (e.g., +, -) are simply extensions of the operators for FIXED and FLOAT numbers, while others (e.g., DOT, CROSS) apply only to vectors and matrices.

The Addition Operator (+)

The expression VA + VB represents the vector sum of two vectors VA and VB, that is, a vector VC in which VCi = VAi + VBi. Similarly, the expression MA + MB is a matrix MC in which MCi,j = MAi,j + MBi,j. In both cases, the sizes of the operands must agree, i.e., VA and VB must both be N-vectors, and MA and MB must both be MxN matrices; the size of the result agrees with the size of the operands.

The Subtraction Operator (-)

The expression VA - VB represents the difference of the vectors VA and VB, that is, a vector VC in which VCi = VAi - VBi. The expression MA - MB yields another matrix MC in which MCi,j = MAi,j - MBi,j. As with addition, the sizes of the operands and the result must agree.

The Negation Operator (-)

The expression -V is the negation of the vector V, that is, a vector whose elements are the negative of the components of V. Similarly, the expression -M represents the element by element negation of the matrix M.

The Multiplication Operator (*)

The multiplication operator * has a number of different interpretations, depending on the data-type of its two operands:

Table 4-1: VECTOR and MATRIX Operations

Form	Type of Result	Description of Result
V1 + V2	VECTOR	Vector addition, sizes must agree.
M1 + M2	MATRIX	Matrix addition, sizes must agree.
- V	VECTOR	Negation of a vector.
- M	MATRIX	Negation of a matrix.
V1 - V2	VECTOR	Vector subtract; sizes must agree.
M1 - M2	MATRIX	Matrix subtract; sizes must agree.
F * V V * F	VECTOR	Scalar multiplication of vector.
F * M M * F	MATRIX	Scalar multiplication of matrix.
V * M	VECTOR (ROW)	Pre-multiplication of matrix by row vector; sizes must agree.
M * V	VECTOR (COLUMN)	Post-multiplication of matrix by column vector, sizes must agree.
M1 * M2	MATRIX	Matrix multiplication, M1 is NxM M2 is MxQ, result is NxQ.
V1 DOT V2 V1 * V2	FLOAT	Dot (inner) product of vectors; sizes must agree.
V1 CROSS V2 (2-vectors)	FLOAT	Cross product of 2-vectors.
V1 CROSS V2 (3-vectors)	VECTOR	Cross product of 3-vector; result is 3-vector.

Notation Conventions

V,V1,V2	Vectors
M,M1,M2	Matrices
F	Scalar (FLOAT)

1. VA * VB (where the sizes of the N-vectors VA and VB agree) results in a scalar (FLOAT) which is the inner, or dot product of VA and VB:

   $$\mathrm{VA} * \mathrm{VB} = \sum_{i=1}^{N} \mathrm{VAi} * \mathrm{VBi}$$

2. The expression VA * MB (where VA is an N-vector and MB is an NxM matrix) results in an M-vector VC which is the product of premultiplying MB by the (row) vector VA:

   $$\mathrm{VCj} = \sum_{i=1}^{N} \mathrm{VAi} * \mathrm{MBi,j}$$

3. The expression MA * VB (where MA is an NxM matrix and VB is an M-vector) results in an N-vector VC which is the product of postmultiplying MB by the (column) vector VB:

   $$\mathrm{VCi} = \sum_{j=1}^{M} \mathrm{MAi,j} * \mathrm{VBj}$$

4. The expression MA * MB (where MA is an NxM matrix and VB is an MxQ matrix) results in an NxQ matrix product MC:

   $$\mathrm{MCi,j} = \sum_{k=1}^{M} \mathrm{MAi,k} * \mathrm{MBk,j}$$

5. The expressions F * V and V * F are the scalar product of the vector V and the FLOAT number F; that is, a vector VB such that VBi = F * Vi. Similarly, the expressions F * M and M * F are the scalar product of the matrix M and the FLOAT number F, that is, a matrix MB such that MBi,j = F * Mi,j.

Note that the division operator is not defined for vector or matrix operands. To divide each element of a vector or matrix by a scalar, you must multiply by the reciprocal of the scalar.

The DOT Operator

The expression VA DOT VB represents the inner (dot) product of the N-vectors VA and VB (whose sizes must agree), that is, a number:

$$\mathrm{VA\ DOT\ VB} = \sum_{i=1}^{N} \mathrm{VAi} * \mathrm{VBi}$$

The expressions VA DOT VB and VA * VB are equivalent, but the former may be more indicative of the intent.

The CROSS Operator

The result of the CROSS operator depends on the size of the vectors to which it is applied. If VA and VB are 2-vectors, then VA CROSS VB yields a FLOAT number:

```
VA(1) * VB(2) - VB(1) * VA(2)
```

If VA and VB are both 3-vectors, then VA CROSS VB yields a 3-vector VC which is the <u>vector product</u> of VA and VB:

```
VC(1) = VA(2)*VB(3) - VA(3)*VB(2);
VC(2) = VA(3)*VB(1) - VA(1)*VB(3);
VC(3) = VA(1)*VB(2) - VA(2)*VB(1);
```

The expression VA CROSS VB is undefined if VA and VB are not both 2-vectors or 3-vectors.

Precedence of Operators

The precedences (order of evaluation) for the operators +, -, and * are the same for vectors and matrices as for FLOATs and FIXEDs (see Chapter I of the User's Guide to PL/PROPHET). The precedences for DOT and CROSS are the same as for vector multiplication; all operators are left-associative, e.g.,

```
VA * VB * VC  is equivalent to  (VA * VB) * VC
```

Note that the * operator, when used with a matrix operand, is emphatically <u>not</u> commutative; that is, in general, MA*MB <> MB*MA.

Part 3: Vector and Matrix Functions

Other vector and matrix operations are implemented as the function procedures described in Table 4-2. These procedures are called with one or more vector and/or matrix arguments, and return a scalar, vector or matrix result.

MAG,
%MAGSQD

The expression MAG(V) returns the scalar magnitude, or length of the vector V, that is, a FLOAT number such that:

$$MAG(V) = \sqrt{\sum_{i=1}^{N} Vi * Vi}$$

The expression %MAGSQD(V) returns the square of the magnitude of V; it is much faster than MAG, since it does not extract the square root.

%UNITVEC

The expression %UNITVEC(V) returns a unit VECTOR pointing in the same direction as the vector V, that is, a vector whose magnitude is 1.0 and whose elements are proportional to those of V. An important exception occurs if the magnitude of V is zero: in that case, %UNITVEC(V) is a vector the same size as V with all 0.0 elements (effectively, a copy of V).

%PERPTO

The expression %PERPTO(V), where V is a N-vector, returns another N-Vector VP such that V and VP are perpendicular (V DOT VP = 0.0) and VP is non-zero (MAG(VP) > 0.0). VP is only guaranteed to have these two properties, and is otherwise somewhat randomly chosen. One exception can occur: if V is a non-zero 1-vector, the vector [0.] is returned, violating the "non-zero" property.

%TRANSPOSE

The expression %TRANSPOSE(M) returns a MATRIX which is the transpose of the matrix M; that is, if M is a PxQ matrix, then %TRANSPOSE(M) returns a QxP matrix MT such that MTi,j = Mj,i.

%INVERSE

The expression %INVERSE(M) returns a MATRIX which is the multiplicative inverse of the matrix M; that is, if M is an NxN matrix, then %INVERSE(M) returns an NxN matrix MI such that M * MI = MI * M = NxN identity matrix (1's on the diagonal, 0's elsewhere). Several exceptions can occur: if M is not a square matrix, %INVERSE aborts. If M is non-invertible (determinant = 0), then the result returned is an EMPTY matrix to signal this condition.

Table 4-2: VECTOR and MATRIX Functions

Form	Type of Result	Description of Operation
MAG(V)	FLOAT	Magnitude of vector.
%MAGSQD(V)	FLOAT	Square of magnitude of vector; (faster than MAG).
%UNITVEC(V)	VECTOR	Unit vector along V.
%PERPTO(V)	VECTOR	Vector perpendicular to V.
%TRANSPOSE(M)	MATRIX	Transpose of matrix.
%INVERSE(M)	MATRIX	Inverse of (square) matrix.
%DET(M)	FLOAT	Determinant of (square) matrix.
%ANGLE3(P1,P2,P3)	FLOAT	Angle (degrees) formed by three points (2D or 3D).
%ANGLE4(P1,P2,P3,P4)	FLOAT	Torsion angle (degrees) formed by four points (3D).
CALL %MATDIMEN(M,NR,NC)	-----	NR, NC set to number of rows and cols in M, or EMPTY if M is not rectangular.
%IDENTITYMATRIX(N)	MATRIX	NxN identity matrix.
%TRIPOD (V1,V2,A13,A23)	VECTOR	Vector that is A13 degrees from V1 and A23 degrees from V2.

Notation Conventions

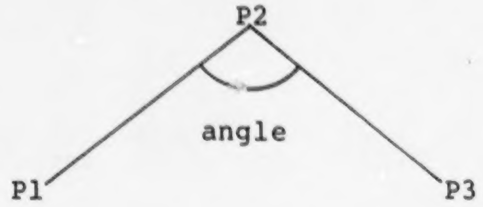
V,V1,V2	VECTORs
M,M1,M2	Matrices
P1,P2,P3,P4	Points (VECTORs)
N,NR,NC	FIXED
A13,A23	FLOAT angles (degrees)

%DET

The expression %DET(M), where M is an NxN (square) matrix, returns the determinant of M.

%ANGLE3

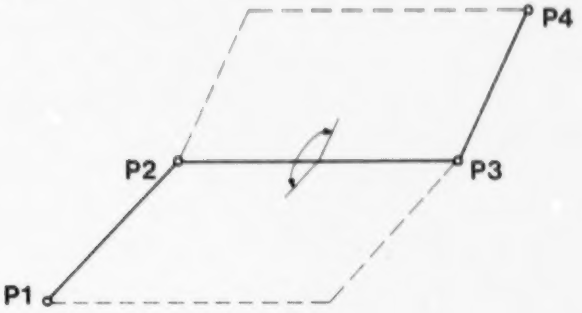
The expression %ANGLE3(P1,P2,P3) returns a FLOAT value which is the number of degrees (0. to 180.) in the angle formed by the (2- or 3D space) points P1, P2, and P3, thus:



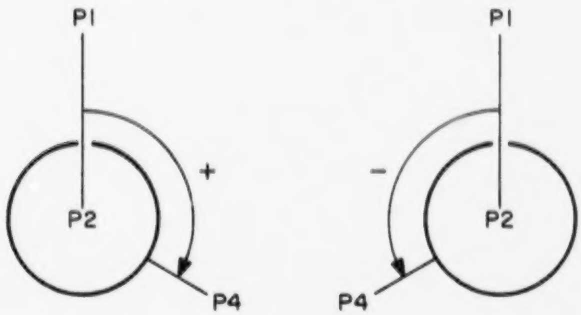
Each "point" is a 2-VECTOR or a 3-VECTOR. %ANGLE3 returns 0 degrees if P1 = P2 or P2 = P3 or P1 = P3.

%ANGLE4

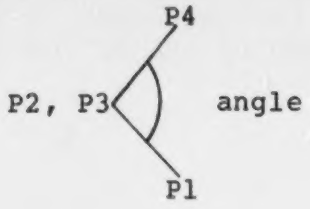
The expression %ANGLE4(P1,P2,P3,P4) yields a FLOAT which is the number of degrees (-180. to 180.) in the torsion angle formed by the points (3-vectors) P1, P2, P3, and P4, that is, the angle between the planes defined by P1,P2,P3 and by P2,P3,P4, thus:



The sign convention is that used by stereochemists, as illustrated in the following Newman projections:



If P2 and P3 are coincident, the torsion angle is not well defined; in that case, the angle between the lines P2-P1 and P2-P4 is returned, thus:



If P1-P2-P3 are colinear, or if P2-P3-P4 are colinear, the torsion angle is undefined, but 0. degrees is returned.

%MATDIMEN The procedure %MATDIMEN can be used to determine the size of a matrix through the statement:

```
CALL %MATDIMEN(M,NR,NC)
```

Where M is a MATRIX, and NR and NC are FIXEDs. Upon return, NR and NC are set to the number of rows and columns in M. However, if M is not a rectangular matrix, then NR and NC will be set to EMPTY.

%IDENTITYMATRIX

The expression %IDENTITYMATRIX(N), where N is a FIXED, yields an NxN <u>identity</u> <u>matrix</u> of the form:

$$\begin{bmatrix} 1. & 0. & 0. & \dots & 0. \\ 0. & 1. & 0. & \dots & 0. \\ 0. & 0. & 1. & \dots & 0. \\ & & \dots & & \\ 0. & 0. & 0. & \dots & 1. \end{bmatrix}$$

An EMPTY matrix is returned if N < 1.

%TRIPOD The expression %TRIPOD(V1,V2,A13,A23), where V1 and V2 are VECTORS and A13 and A23 are FLOAT variables, returns a VECTOR that forms an angle of A13 degrees with vector V1 and an angle of A23 degrees with vector V2.

Definition of Transformers

One of the major uses of vectors and matrices within the PROPHET system procedures is in computations involving three-dimensional (3D) geometry. A 3-vector [X Y Z] can represent a point in 3D space with cartesian coordinates X, Y, and Z. The vector [X Y Z] can also represent the direction of a set of lines in 3D space, namely all lines parallel to one drawn from the origin to the point [X Y Z].

Often, in geometric computations, one wishes to transform points in 3D space by shifting them (*translating*); by *rotating* them about some axis; by shrinking or expanding space about some point (*scaling*); or by combinations of these transformations. PL/PROPHET provides a convenient and efficient way to transform points in 3D space: a *transformer* is a MATRIX which describes a particular transformation in 3D space. PL/PROPHET provides two sets of tools for working with transformers: first, a means of applying transformers to points (3-vectors) to transform them in 3D space, and second, a means of generating and combining transformers to perform particular kinds of tranformations.

The XFM Operator

The expression V XFM X, where V is a 3-vector and X is a transformer (MATRIX), yields another 3-vector having the following characteristics: if V represents a point in 3D space, and X represents a transformation, then V XFM X represents the new point which results from performing the specified transformation (movement) on the specified point. For example, if X = %TRANSLATION(3.,4.,5.) (see below), then V XFM X is a point which results from moving point V by 3 units in the X-direction, 4 units in the Y-direction, and 5 units in the Z-direction.

Combining Transformers

Sometimes one wishes to apply several tranformations to one or more points. One approach is to define several transformers and apply them in succession. For example, suppose X1 is a translation and X2 is a rotation; to achieve the effect of translating, then rotating a point V, one could write:

```
(V XFM X1) XFM X2
```

Table 4-3: Transformer Operations and Procedures

Form	Type of Result	Description of Result
V XFM T	VECTOR	Transformation of vector.
T1 XFM T2	TRANSFORMER	Composition of transformations.
%TRANSLATION(X,Y,X)	TRANSFORMER	Transformer to translate by X,Y,Z.
%TRANSLATEV(V)	TRANSFORMER	Transformer to translate by V(1).
%ROTATION(VC,VA,A)	TRANSFORMER	Transformer to rotate about axis vector VA, center VC, by A degrees.
%ROTATEONLY(T)	TRANSFORMER	Transformer containing only rotation components of T.
%SCALE(SX,SY,SZ)	TRANSFORMER	Transformer to scale by factors SX, SY, and SZ about the origin.
%SCALEV(V)	TRANSFORMER	Transformer to scale by V(1), V(2), and V(3) about the origin.
%SCALEFAC(T)	VECTOR	Scale factors in T for X, Y and Z.
%SCALEABOUT(SX,SY, SZ,CX,CY,CZ)	TRANSFORMER	Transformer to scale by SX, SY, SZ about center CX, CY, CZ.
%REFLECTION(VC,VN)	TRANSFORMER	Transformer to reflect through plane containing VC, normal VN.
%IDENTITYXFM	TRANSFORMER	Identity transformer.
%INVXFM(T)	TRANSFORMER	Inverse of transformer ("undoes T").
%CRYSTXFM(A,B,C,ALPHA, BETA,GAMMA)	TRANSFORMER	Transformer to convert crystal coordinates to cartesian, based on cell constants.

Notation Conventions

V,VA,VC	VECTORs
T,T1,T2	Transformers (MATRIXes)
A,SX,SY,SZ	Scalars (FLOATs)
X,Y,Z	Scalars (FLOATs)
CX,CY,CZ	Scalars (FLOATs)
B,C,ALPHA	Scalars (FLOATs)
BETA,GAMMA	Scalars (FLOATs)

A different approach is to recognize that "translating, then rotating" is itself a transformation, which can be represented by a transformer. Thus, in PL/PROPHET, the expression X1 XFM X2 yields a transformer which has the same effect as the successive application (composition) of X1 followed by X2. That is,

```
(V XFM X1) XFM X2  =  V XFM (X1 XFM X2)
```

Thus, one may write V XFM X1 XFM X2 without regarding the order of evaluation.

In situations where a combination of transformers must be applied many times (e.g., to an array of points), it is often convenient to define a temporary variable to hold the combined transformer, thus:

```
/* XFMALL applies transformer X1, then transformer  X2  to
   an array of points PA, to yield another array TPA */

      XFMALL:  PROCEDURE(PA,X1,X2,TPA);

      DECLARE  /* Arguments */
       PA  VECTOR ARRAY,    /* Untransformed pts. [IN] */
       (X1,X2) MATRIX,      /* Transformers [IN] */
       TPA VECTOR ARRAY;    /* Transformed pts. [OUT] */

      DECLARE  /* Locals */
       X1THENX2    MATRIX, /* Combined trans. */
       I       FIXED;  /* Loop index */

       X1THENX2 = X1 XFM X2;    /* Combination */

       DO I = 1 TO TOP(PA);     /* Apply it to pts. */
           TPA(I) = PA(I) XFM X1THENX2;
       END;     /* DO I */

      END; /* XFMALL */
```

Notice that in the cases above where transformers are associated with arguments (X1, X2) or local variables (X1THENX2), they are declared as matrices.

Transformer-Building Procedures

To facilitate constructing transformers, a number of procedures return transformers to perform several classes of geometrically meaningful transformations in 3D space (see Table 4-3). Remember that transformers are of type MATRIX, and must be declared if they are to be stored in variables or passed as arguments.

%SCALE, %SCALEV

The expression

T = %SCALE(SX,SY,SZ)

where SX, SY, and SZ are FLOATs, yields a transformer (T in this example) which has the effect of scaling 3D space by SX in the x-direction, SY in the y-direction, and SZ about the z-direction, about the origin. For example, if V is the vector [Vx Vy Vz], then V XFM %SCALE(SX,SY,SZ) yields the vector [SX*Vx SY*Vy SZ*Vz]. The function

T = %SCALEV(V)

differs from %SCALE only it that it takes a single argument, which is a vector whose first three components are used as SX, SY, and SZ.

%SCALEABOUT

The expression

T = %SCALEABOUT(SX,SY,SZ,CX,CY,CZ)

yields a transformer which scales by SX, SY, and SZ about the point [CX CY CZ]. This differs from %SCALE, which scales about the origin; that is,

%SCALE(SX,SY,SZ)

is the same as

%SCALEABOUT(SX,SY,SZ,0.,0.,0.)

%SCALEFAC

The expression

V = %SCALEFAC(T)

returns a 3-vector containing the scale factors in the X, Y and Z directions (SX, SY and SZ in the examples above) found in the transformer T.

%TRANSLATION, %TRANSLATEV

The expression

```
T = %TRANSLATION(TX,TY,TZ)
```

yields a tranformer which <u>translates</u> (shifts) points by TX in the x-direction, TY in the y-directions, and TZ in the z-direction. The expression

```
T = %TRANSLATEV(V)
```

takes a single argument, a vector whose first three elements specify the translations.

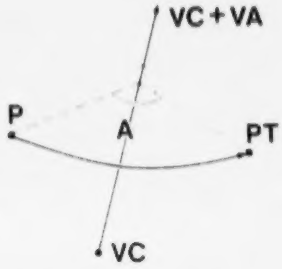
%ROTATION

The expression

```
T = %ROTATION(VC,VA,A)
```

yields a transformer which rotates points in 3D space A degrees around an axis having direction VA (a 3-vector), passing through point (3-vector) VC. For example,

```
PT = P XFM %ROTATION(VC,VA,A)
```



If VA is a zero-vector (MAG(VA) = 0.) then %ROTATION aborts.

%ROTATEONLY

The expression

```
T2 = %ROTATEONLY(T1)
```

returns a transformer (T2 in this example) that contains only the rotational components of the argument transformer (T1 in the example).

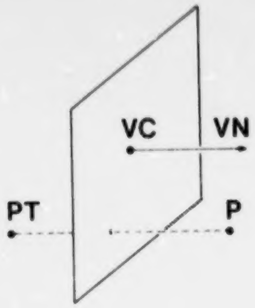
%REFLECTION

The expression

```
T = %REFLECTION(VC,VN)
```

yields a transformer which performs a <u>reflection</u> of points through a mirror plane defined by the 3-vectors VC and VN: VC is a point contained in the mirror plane, and VN is the direction of the normal (perpendicular vector) to the plane. For example:

`PT = P XFM %REFLECTION(VC,VN)`



%INVXFM The expression

`T = %INVXFM(X)`

yields a transformer which is the inverse of the transformer X; that is, it "undoes" the effect of X. Thus, for all 3-vectors V:

`V XFM (X XFM %INVXFM(X)) = V XFM (%INVXFM(X) XFM X) = V`

The procedure aborts if the argument is not a transformer (4x3 matrix), or if the transformation X is non-invertible. This can arise only if X is a scaling transformation (%SCALE, %SCALEV) in which one or more scale factors was 0., or if X was created by combining some other non-invertible transformer using XFM.

%IDENTITYXFM The expression

`T = %IDENTITYXFM`

returns the <u>identity transformer</u>, that is, the transformer that results in no change to the vector or matrix to which it is applied.

%CRYSTXFM The expression

`T = %CRYSTXFM(A,B,C,ALPHA,BETA,GAMMA)`

can be used to convert coordinates obtained for atoms in a molecule through crystallographic measurements into cartesian coordinates. If A,B,C,ALPHA,BETA and GAMMA are the crystallographic cell constants, the matrix returned contains the transformation that must be applied to achieve this conversion.

Transformer Implementation Details

The remainder of this part describes in some detail the implementation of transformers in PL/PROPHET, and may be of interest only to those curious about these internal operations.

Any similarity (shape-preserving) transformation in 3D space can be achieved by an isotropic scaling about the origin, followed by a rotation about the origin, followed by a translation. Formally:

$$PT = (S * P) * R + T$$

where:

P is the original point (3-vector)
S is a scale factor (FLOAT)
R is a 3x3 rotation matrix
T is a translation (3-vector)
PT is the transformed version of P

In PL/PROPHET, such a transformation is represented by a transformer, a 4x3 matrix:

$$\begin{bmatrix} S*R(1,1) & S*R(1,2) & S*R(1,3) \\ S*R(2,1) & S*R(2,2) & S*R(2,3) \\ S*R(3,1) & S*R(3,2) & S*R(3,3) \\ T(1) & T(2) & T(3) \end{bmatrix}$$

or, in formal notation:

$$\left[\begin{array}{c} S * R \\ \hline T \end{array}\right]$$

The XFM operator transforms a point (3-vector) P by multiplying P by the first three rows of the transformer (S*R) and adding the fourth row (T) to the result.

The XFM operator is also used to compute the composition of two transformers, thus:

transform1: $PT = (S * P) * R + T$
transform2: $PT' = (S' * P') * R' + T'$

transform1
followed by: $PT' = S'*S*P*R*R' + S'*T*R' + T'$
transform2

In that case, the transformer for the composition is defined (formally) to be:

$$\left[\begin{array}{c} S' * S * R * R' \\ \hline S' * T * R' + T' \end{array}\right]$$

The inverse of a transformation can be computed as follows:

$$PT = (S * P) * R + T$$

$$P = 1/S * (PT - T) * R^{-1}$$

$$= 1/S * PT * R^{-1} - 1/S * T * R^{-1}$$

Thus, the inverse of a transformer (%INVXFM) can be written:

$$\text{\%INVXFM} \qquad \left[\begin{array}{c} S * R \\ \hline T \end{array}\right] = \left[\begin{array}{c} 1/S * R^{-1} \\ \hline -1/S * T * R^{-1} \end{array}\right]$$

Part 5: Examples of VECTOR and MATRIX Uses

This part presents several examples of how the PL/PROPHET VECTOR and MATRIX facilities might be used.

Point Spread

It is sometimes useful to have a measure of the spread of a set of points, whether they are in a plane or in 3D-space. One possible definition of the spread is suggested by analogy with the standard deviation of a set of scalars: the spread of a set of points is the root-mean-square distance of the points from the "mean point". The latter can also be defined by analogy to the mean of a set of scalar points. Thus:

$$\text{MEAN:} \quad \bar{V} = \frac{1}{N} \sum_{i=1}^{N} Vi$$

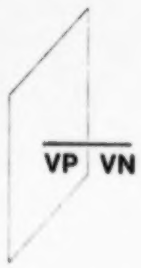
$$\text{SPREAD} = \frac{1}{N} \sqrt{\sum_{i=1}^{N} \left| (Vi - \bar{V}) \right|^{2}}$$

The procedure on the next page computes the spread of a set of points, expressed as an array of vectors. Notice that no mention of the size of the vectors is made, so the procedure can work in an arbitrary N-dimensional space.

```
/* PTSPRD RETURNS SPREAD OF A SET OF N-SPACE POINTS */

   PTSPRD: PROCEDURE(PTS);
    RETURNS FLOAT;

   DECLARE      /* Arguments */
    PTS VECTOR ARRAY;   /* Set of points [IN] */

   DECLARE      /* Locals */
    N   FIXED,  /* No. of pts. */
    I   FIXED,  /* Index */
    SUMPT   VECTOR, /* Sum of pts. */
    AVGPT   VECTOR, /* Avg. pt. */
    SUMD2   FLOAT;  /* Sum dists**2 */

    N = TOP(PTS);
    IF (N = 0) THEN RETURN(0.);

   /* Compute average point */

    SUMPT = PTS(1);
    DO I = 2 TO N;  /*Sum up pts */
        SUMPT = SUMPT + PTS(I);
    END;    /* DO I */

    AVGPT = (1.0/CFLOAT(N)) * SUMPT;
   /* Compute sum of squares of dists. to avg. point */

    SUMD2 = 0.0;
    DO I =1 TO N;
        SUMD2 = SUMD2 + %MAGSQD(PTS(I)-AVGPT);
    END;    /* DO I */

   /* Return root mean square */
    RETURN (SQRT(SUMD2/CFLOAT(N)));
   END; /* PTSPRD */
```

Plane Manipulation

Many geometrical computations involve descriptions of planes in 3D-space. One method of representing planes is by a pair of vectors, VP and VN, where VP is a point contained in the plane and VN is a unit vector in a direction perpendicular to the plane, thus:



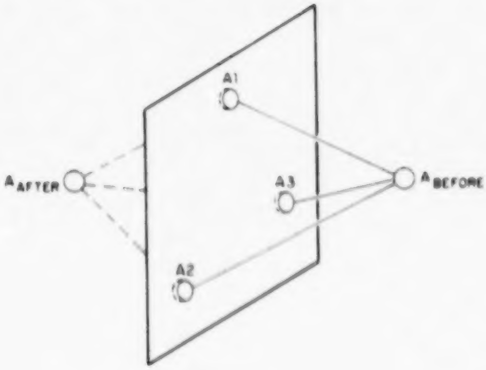
The following routines work with this VP,VN description of planes:

```
/* PLANENORM returns a 3-vector which is the normal to the
   plane formed by the three argument points (3-vector) */

   PLANENORM: PROCEDURE(P1,P2,P3);
    RETURNS VECTOR;

   DECLARE      /* Arguments */
    (P1,P2,P3) VECTOR;  /* 3 points [in] */

   DECLARE      /* Locals */
    VN  VECTOR; /* Normal vec. */

   /* Compute unit normal vector; return if non-zero,
      else return empty */

    VN = %UNITVEC ((P2 -P1) CROSS (P3 - P2));
    IF (%MAGSQD(VN) = 0.0) THEN CLEAR VN;
    RETURN (VN);

   END; /* PLANENORM */
```

```
/* PLANEDIST returns the (signed) distance from a point to
   a plane described by the two vectors */

      PLANEDIST: PROCEDURE(PT, PLP, PLN);
       RETURNS FLOAT;

      DECLARE       /* Arguments */
       PT VECTOR,   /* The point [IN] */
       PLP VECTOR,  /* Pt. in the plane [IN] */
       PLN VECTOR;  /* Unit normal to plane [IN] */

      RETURN ((PT - PLP) DOT PLN);

      END; /* PLANEDIST */


/* PLANEPROJ returns the projection of a point on a  plane
   described by the two vectors */

      PLANEPROJ: PROCEDURE (PT, PLP, PLN);
       RETURNS VECTOR;

      DECLARE       /* Arguments */
       PT  VECTOR,  /* The point  [IN] */
       PLP VECTOR,  /* Pt. in plane [IN] */
       PLN VECTOR;  /* Unit normal to plane [IN] */

      DECLARE       /* Locals */
       D   FLOAT;   /* (Signed) dist. to PLN */

       D = PLANEDIST (PT, PLP, PLN);

       RETURN (PT - D*PLN);

      END; /* PLANEPROJ */
```

Reflecting a Molecule

The last example presents the use of a transformer to alter the coordinates of a molecule. Specifically, it reflects the coordinates through a mirror plane which is defined by three of the atoms in the molecule, thus:



(Note that REFLECTMOL uses some molecule functions which are described in Chapter III above.)

```
/* REFLECTMOL reflects the stored model coordinates of its
   argument  molecule  through a plane determined by three
   atoms, whose indices are the 2nd  thru  4th  arguments.
   Assumes   all   atoms   have  coordinates  which  can  be
   retrieved with %GETMVEC and modified with %SETMVEC.  */

      REFLECTMOL: PROCEDURE(MOLQ,A1,A2,A3);

      DECLARE       /* Arguments */
       MOLQ    MOLECULE,   /* Reflectee [IN] */
       (A1,A2,A3) FIXED;   /* Atom indices [IN] */

      DECLARE       /* Locals */
       (VM1,VM2,VM3)   VECTOR, /* Atom locations */
       PLN VECTOR, /* Vector normal to plane */
       RFLXFM  MATRIX, /* Transformer */
       I   FIXED,  /* Loop index */
       VM  VECTOR; /* Atom loc */

      /* Retrieve locations of atoms defining the
         reflection plane */

       VM1 = %GETMVEC(MOLQ,A1);
       VM2 = %GETMVEC(MOLQ,A2);
       VM3 = %GETMVEC(MOLQ,A3);

      /* Compute normal to reflection plane; if empty
         exit without doing anything */
       PLN = PLANENORM(VM1,VM2,VM3);
       IF (EMPTY(PLN)) THEN RETURN;

      /* Compute reflection transformer */

       RFLXFM = %REFLECTION (VM1,PLN);

      /* Apply transformer to all atom locations */

       DO I = 1 TO %LASTATOM(MOLQ);
           VM = %GETMVEC(MOLQ,I);
           CALL %SETMVEC(MOLQ,I,VM XFM RFLXFM);
       END;    /* DO I */

      END; /* REFLECTMOL */
```

5. Public Procedures

Chapter V.

PUBLIC MOLECULE PROCEDURES

This chapter describes those PROPHET *Public Procedures* that are designed to be used in conjunction with the PROPHET molecule facilities described in the rest of this manual. These procedures cover a wide range of diverse applications, but they are presented here in strict alphabetical order so that you can easily locate any procedure if you know its name. These procedures may be somewhat arbitrarily broken into five classes, as follows:

1. Procedures that construct new molecules
2. Procedures that modify existing molecules
3. Procedures that manipulate molecular models.
4. Procedures that create special molecule displays.
5. Procedures that measure molecular properties.

The procedures that fall into each of the five categories are given in the following lists (several procedures appear in more than one list):

Procedures That Construct New Molecules

CONNECT
CRYST
MBLD
MODIFY
MOLECYCLE
MOLEDAT
PRECONNECT
TABS2MOL

Procedures That Modify Existing Molecules

CLEAVEMOL
MFBADDBONDS
MODIFY
SETATOMNAMES
SETMOLNAMES
SETPEPTIDEBONDS

Procedures That Manipulate Molecular Models

EDITMODEL
FITMOL
GENMOL
ORMOL
REFINEMOL
SETPEPTIDEBONDS

Procedures That Create Special Molecule Displays

NEWMAN
SPACEMOL
STEREOBOX

Procedures That Measure Molecular Properties

BESTPLANE
CLEAVEMOL
CONNLIST
DISTAB
MAST
MOLEDAT
PLANE

The first portion of each procedure's description contains in succinct format a SUMMARY of what the procedures do; a CALLING SEQUENCE, with the arguments and their data types; and ENTRY points, if any. The CITATION section suggests a format for citing that public procedure; QUESTIONS OR COMMENTS indicates the person to whom any questions about the procedure whould be directed; and THINGS TO REMEMBER tries to alert you to any pitfalls the procedure may have. The summary box will serve as a quick reference to running the procedure, but it should be emphasized that a close reading of the more detailed documentation is essential for most procedures.

The descriptions of many of the procedures include examples of their operation. Where the procedure uses an input table, there is usually a public table available (the name is that shown in the example) so that you may try it by yourself. These tables are normally kept archived, so you will have to dearchive them to run the procedure.

It is important that the operation of a public procedure should always be reflected in its documentation. Consequently, no one should make any change in the coding of a procedure stored in the Public Saved Table. You may, of course, make a private copy of any public procedure, and change it to meet your specific needs. Please do contact your PROPHET representative if you feel changes should be made in the version of any public procedure saved in the Public Saved Table.

The entire user community is indebted to those users who have so generously shared their programming skills by making their procedures public. The molecule user community is particularly indebted to the PROPHET users at the Medical Foundation of Buffalo, who have contributed a great many public procedures and created numerous public tables containing crystallographic molecular information (see Appendix B). In fact, a number of the public procedures they have contributed have recently found their way into the main PROPHET system itself as system procedures (see, for example, the atom name gatekeepers described in Chapter III, Part 2).

In some cases, we have made some minor changes to public procedures donated by users in order to achieve a greater degree of consistency of terminology, or to provide for easy recovery from errors. If you find bugs or other problems in any of the public procedures, please bring them to the attention of the person indicated under QUESTIONS OR COMMENTS in the summary box.

If you have written a procedure that you think other PROPHET users might find useful, please send it to your system representative, _along with_ its documentation and examples of its operation. Please refer to the _Public Procedures Notebook_ for some suggested guidelines for writing and documenting public procedures.

BESTPLANE

by Mark Lavin
Bolt Beranek and Newman Inc.

Summary

BESTPLANE takes an array of 3-component vectors (corresponding to points in 3-space) and computes the equation of the best fit plane through those points, that is, the plane which minimizes the mean squared distance to each point. BESTPLANE returns a 4-component vector [A1, A2, A3, D], such that [A1, A2, A3] is normal to the best fit plan and $A_1X+A_2Y+A_3Z+D = 0$ is the equation of the best fit plane.

Calling Sequence

```
RAC PUBLIC BESTPLANE

BESTFIT = BESTPLANE(POINTS)
<vector>         <vector-array>
                       or
                    <matrix>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

The following example illustrates the use of BESTPLANE. Suppose you wanted to type the distances of all atoms in molecule M from a best fit plane through M.

The following procedure does just that:

BESTPLANE-2

```
PROCEDURE(M);
DECLARE
     M           MOLECULE,      /* MOLECULE [IN] */
     MODCOOR     MATRIX,        /* MODEL COORDINATES OF
                                   EACH ATOM */
     BESTFIT     VECTOR,
     A           VECTOR,        /* NORMAL TO BESTFIT PLANE */
     D           FIXED,         /* OFFSET OF BESTFIT PLANE */
     I           FIXED;         /* LOOP INDEX */

             RAC PUBLIC BESTPLANE;
     /* SAVE ALL MODEL COORDINATES IN MODCOOR */
     DO I=1 TO %LASTATOM(M);
        MODCOOR[I] = %GETMVEC(M,I);
     END;
        /* COMPUTE BESTFIT PLANE */

     BESTFIT = BESTPLANE(MODCOOR);
     A[1] = BESTFIT[1];         /* direction of normal */
     A[2] = BESTFIT[2];
     A[3] = BESTFIT[3];
     D = BESTFIT[4];
          /* TYPE SIGNED DISTANCES FROM BESTFIT PLANE */

     DO I = 1 TO %LASTATOM(M);
     TYPE (MODCOOR[I] DOT A) + D;
     END;

END;
```

CLEAVEMOL

by Mark A. Lavin
Bolt Beranek and Newman, Inc.

Summary

CLEAVEMOL determines the set of components into which a molecule would be broken by removal of a specified set of bonds.

Calling Sequence

```
RAC PUBLIC CLEAVEMOL

CALL CLEAVEMOL(mol-name, cleave-bond-list,
               <molecule>   <fixed array>

               ordered-bond-list, component-atom-lists
                 <fixed array>    <fixed array array>

               error-text)
                 <text>
```

Citation

No citation necessary.

Questions and Comments

Contact Wayne Rindone on the PROPHET line.

Things to remember

CLEAVEMOL works on all types of terminals.

CLEAVEMOL determines the set of components into which a molecule would be broken by removal of a specified set of bonds. Specifically, you supply as input arguments a molecule and a set of bonds, represented by a FIXED ARRAY of their numeric indices, "cleave-bond-list". CLEAVEMOL creates in its output arguments lists of the set of components; these are represented as the FIXED ARRAY ARRAY "component-atom-lists", in which component-atom-lists[i,j] is the numeric index of the j-th atom in the i-th component into which the molecule is broken. In addition, CLEAVEMOL creates another list of bond indices, represented by the FIXED ARRAY "ordered-bond-list", which is the original set of bonds, sorted in order of occurrence in the structure of the molecule M. That is, ordered-bond-list[j] is the bond between component "j" and component "j+1". This implies an important restriction on the applicability of CLEAVEMOL: the set of components into which the molecule is divided must be linearly ordered in the structure of the molecule.

The original set of partitioning bonds may be presented in any order; the "ordered bond list" is always "returned" in sorted order, although there is no control over whether the sorted order is from "right to left" or "left to right". CLEAVEMOL also reports exceptions through the output TEXT variable "error-text"; "error-text" is EMPTY if there are no exceptions, 'RING' if one of the cleaving bonds is part of a ring or ring system, and 'BRANCH' if the set of components form a branching, rather than a linear, structure.

Two uses have been suggested for CLEAVEMOL. In one case, it has been used to partition a molecule before invoking of the CAMSEQ energy minimizer, which requires that the molecule be broken up into the components separated by the bonds about which to minimize torsional potential. Another use is to identify the components of a polymer given the inter-monomeric bonds; you could mark the atoms in each amino acid in a polypeptide, given knowledge of where the peptide bonds are (cf. Public Procedure SETPEPTIDEBONDS).

CLEAVEMOL employs an iterative graph-walking procedure to mark atoms and bonds. Cursory analysis of the algorithm suggests a running time on the order of:

$$k1 * (A+B) * (1 + k2*C)$$

where A, B, and C are the number of atoms, bonds, and cleaving bonds in the molecule, and k1 and k2 are arbitrary constants.

Figures 1 - 6 present an example of the use of CLEAVEMOL. In this example, we wish to selectively display the subunits of a pentaglycine molecule whose conformation has been calculated by the COMPUTE MODEL command afer using the public procedure SETPEPTIDEBONDS. In Figure 1 we display the molecule together with bond indices to determine which bonds separate the amino acid subunits.

```
>DIS PENTAGLYCINE THREED BONDINDEX< >EA<
```

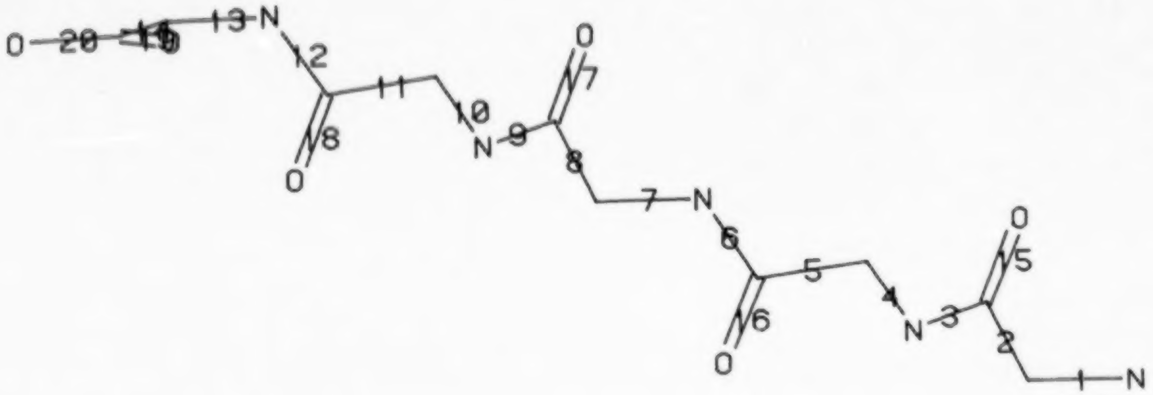


Figure 1. PENTAGLYCINE molecule displayed THREED BONDINDEX.

Once we have observed that the peptide bonds have indices 3,6,9 and 12, we can use CLEAVEMOL to determine the indices of the atoms in each subunit, as illustrated in Figure 2. After returning from CLEAVEMOL, we check that the argument called ERRORTEXT has not been set to indicate any problems, and we examine the ordered list of separating bonds and the five sets of atom indices representing the atoms in each subunit.

In Figure 3, we use the system procedure %MARKATOMS, which is described in Section 3H of Chapter III, to set up selecting columns corresponding to each glycine subunit in PENTAGLYCINE. Figures 4 and 5 show the state of the selecting columns in the ATOMS table of the PENTAGLYCINE molecule at this point. Finally, in Figure 6, we use the selective display feature described in Part 3 of Chapter II to display the first and fourth glycine subunits in PENTAGLYCINE.

```
>RAC PUBLIC CLEAVEMOL< >DECLARE ORDEREDBONDS FIXED ARRAY< >DECLARE
 ATOMLISTS FIXED ARRAY ARRAY< >DECLARE ERRORTEXT TEXT< >CALL CLEAVEMOL(P
##ENTAGLYCINE,[3,6,9,12],ORDEREDBONDS,ATOMLISTS,ERRORTEXT)< >TYPE
 ERRORTEXT< UNDEFINED >DISPLAY ORDEREDBONDS< >DIS ATOMLISTS< >EA<

   4R X 1C
      |
      | 1
  ----+----
   1  | 12
   2  | 9
   3  | 6
   4  | 3

   5R X 9C
      |
      | 1   2   3   4   5   6   7   8   9
  ----+------------------------------------
   1  | 13  14  15  20  21  22  23  24  25
   2  | 10  11  12  19  26  27  28
   3  | 7   8   9   18  29  30  31
   4  | 4   5   6   17  32  33  34
   5  | 3   2   1   37  38  35  36  16
```

Figure 2. Example of calling CLEAVEMOL to determine amino acid subunits in PENTAGLYCINE.

```
>CALL %MARKATOMS(PENTAGLYCINE,'GLY5',ATOMLISTS[1])<
ADDED COLUMN  GLY5  TO ATOMS TABLE.
>CALL %MARKATOMS(PENTAGLYCINE,'GLY4',ATOMLISTS[2])<
ADDED COLUMN  GLY4  TO ATOMS TABLE.
>CALL %MARKATOMS(PENTAGLYCINE,'GLY3',ATOMLISTS[3])<
ADDED COLUMN  GLY3  TO ATOMS TABLE.
>CALL %MARKATOMS(PENTAGLYCINE,'GLY2',ATOMLISTS[4])<
ADDED COLUMN  GLY2  TO ATOMS TABLE.
>CALL %MARKATOMS(PENTAGLYCINE,'GLY1',ATOMLISTS[5])<
ADDED COLUMN  GLY1  TO ATOMS TABLE.
>SCAN ATOMS OF PENTAGLYCINE COLS 1, 8 TO<
```

Figure 3. Use of %MARKATOMS to set up selecting columns for each amino acid subunit in PENTAGLYCINE.

```
DO YOU WISH TO SEE MORE?Y
```

ATOMS 38R X 12C

	1 TYPE	8 GLY5	9 GLY4	10 GLY3	11 GLY2	12 GLY1
1	N					TRUE
2	C					TRUE
3	C					TRUE
4	N				TRUE	
5	C				TRUE	
6	C				TRUE	
7	N			TRUE		
8	C			TRUE		
9	C			TRUE		
10	N		TRUE			
11	C		TRUE			
12	C		TRUE			
13	N	TRUE				
14	C	TRUE				
15	C	TRUE				
16	O					TRUE
17	O				TRUE	
18	O			TRUE		
19	O		TRUE			
20	O	TRUE				

Figure 4. ATOMS OF PENTAGLYCINE.

CLEAVEMOL-6

```
END OF SCAN--HIT [GO] TO PROCEED

ATOMS  38R X 12C
```

	1 TYPE	8 GLY5	9 GLY4	10 GLY3	11 GLY2	12 GLY1
21	O	TRUE				
22	H	TRUE				
23	H	TRUE				
24	H	TRUE				
25	H	TRUE				
26	H		TRUE			
27	H		TRUE			
28	H		TRUE			
29	H			TRUE		
30	H			TRUE		
31	H			TRUE		
32	H				TRUE	
33	H				TRUE	
34	H				TRUE	
35	H					TRUE
36	H					TRUE
37	H					TRUE
38	H					TRUE

Figure 5. ATOMS OF PENTAGLYCINE (cont'd)

```
>DEFAULT MOLECULES TO THREED COMPLETE< >DIS PENTAGLYCINE SELECTING
 ATOMS BY 'GLY1'< >DIS PENTAGLYCINE SELECTING ATOMS BY 'GLY5'< >
```

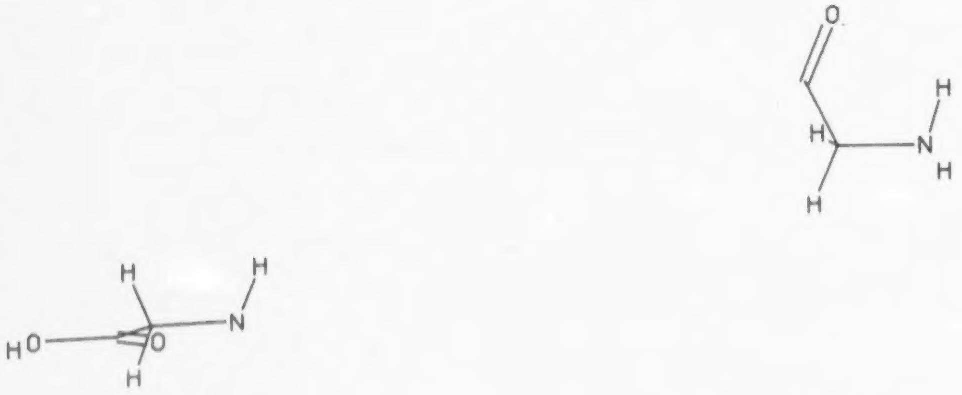


Figure 6. Selective display of subunits 1 and 5 in PENTAGLYCINE.

CONNECT

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo

Summary

CONNECT creates a PROPHET molecule from atom coordinates and information stored in a file or in arrays.

Calling Sequence

```
RAC PUBLIC CONNECT
CALL CONNECT(coord-file)
              <text>
```

Entry

```
CALL MOLCONNECT(mol-text, other-names,  atom-ids,
                <text>    <text-array> <text-array>
                atom-types,   X,      Y,      Z)
                <text-array><float<float<float
                             array>array>array>
```

(See the documentation of CRYST for an alternative way of using CONNECT)

Citation

Potter, S. and Weeks, C.; "CONNECT" in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

Atom names must be provided.

CONNECT works on all types of terminals. Consult Appendix B to see how information is retrieved from a molecule constructed from a coordinate file.

The public procedure CONNECT is an interactive procedure which creates a PROPHET molecule from a coordinate file prepared from 80-column punched cards described below.

The molecules should be continuous. The same enantiomorph should be used for molecules which are to be compared. Atom names should be chosen so as to convey as much information as possible, and logical extensions to IUPAC nomenclature should be used where appropriate. Atoms in similar positions in related structures should have the same names. Hydrogens which are readily distinguished, such as α and β-hydrogens on the steroid nucleus, should be named in accordance with the distinguishing feature.

Format of the Coordinate-File (Numbers in left column refer to line (or card)).

1. Structure name

   col 1 - 8 Structure ID (5 characters maximum, left justified)
   9 - 80 IUPAC name

2. col 9 - 80 IUPAC name, continued, if necessary. Otherwise insert a blank line (card).

3. Reference

   col 1 - 8 Structure ID
   9 - 80 Author, journal, volume, page (year).

4. Cell constants

   FORMATS (6F10,2X,A6,A10)

   col 1 - 10 a
   11 - 20 b
   21 - 31 c
   31 - 40 α
   41 - 50 β
   51 - 60 γ
   63 - 70 Structure ID
   71 - 80 Space group name

5. Equivalent positions (1 card / position)

FORMAT (3(F10,3I5),I3,I2)

The rotation matrix and translation vector for each position should be given. The coordinates, $\underline{X}$, are transformed to the coordinates for the m'th position, $\underline{X}m$, by the equation.

$$\underline{X}m = Rm\underline{X}+\underline{t}m.$$

col	1 - 10 t_1	11 - 15 R11	16 - 20 R12	21 - 25 R13
	26 - 35 t_2	36 - 40 R21	41 - 45 R22	46 - 50 R23
	51 - 60 t_3	61 - 65 R31	66 - 70 R32	71 - 75 R33

col 76 - 78 Space group number (the number of the space group in *International Tables for X-ray Crystallography.)*

79 - 80 Index number for the position.

The positions are terminated by a card with t_1 = 100.

6. Atomic Coordinates

FORMAT (6F10,10X,A10)

X = 50. is used as an end-of-molecule indicator. It should be placed between groups of atoms not joined by covalent bonds.

X = 100. is the end-of-structure indicator. Columns 73 - 80 will be used to form the atom name (blanks will be suppressed).

col	1 - 10	X	fractional coordinates
	11 - 20	Y	
	21 - 30	Z	
	31 - 40	X standard deviation	
	41 - 50	Y standard deviation	
	51 - 60	Z standard deviation	

col 71 - 72	Molecule serial number (used when the structure contains more than one independent molecule).
73 - 74	Atomic symbol (from the periodic table). T is used to indicate a theoretical hydrogen position (i.e. a position not observed by X-ray). This field should be left justified.
75 - 80	Additional characters needed to form the atom name.

Be certain that no two atoms in a molecule have the same name (columns 73 - 80).

Using CONNECT

To use CONNECT, type the command

```
CALL CONNECT(coord-file)
```

where "coord-file" is the TEXT name of the file containing the coordinate and labelling information.

CONNECT creates bonds between atoms by calling the procedure %CONNECTMOL. See the documentation of %CONNECTMOL in Chapter III, part 3, for further explanation on how bonds are assigned. Registry numbers and IUPAC names are read from the coordinate file, and you have the opportunity to supply additional names. Atoms may be added or deleted to the molecules as it is being made from the coordinate file, and bond types may be defined. If the coordinates for more than one covalently bonded unit are included in the coordinate file, the proper molecular number must be supplied.

Using MOLCONNECT

If you have stored X, Y, and Z coordinates in FLOAT arrays, and the atom types and atom names in TEXT arrays, you can use the entry MOLCONNECT to build a PROPHET molecule without having to create a file. The command is

```
CALL MOLCONNECT(mol-text, other-names, atom-ids,
                atom-types, X, Y, Z)
```

where the arguments are:

"mol-text" - the TEXT of the name you want the molecule variable to have

"other-names" - a TEXT array containing additional names for the molecule. If "other-names" is empty, there will be no additional names.

"atom-ids" - a TEXT array containing the names of each atom (e.g., C1, C2, H10A). This argument must not be empty.

"atom-types" - a TEXT array containing the type of each atom (e.g. C, H, O)

X, Y, Z - FLOAT arrays containing the X, Y, and Z coordinates of each atom.

The MOLCONNECT entry creates a molecule from the array data and calls %CONNECTMOL which interactively assigns bonds and finishes the construction of the molecule.

See the public procedure CRYST for an alternative to MOLCONNECT.

CONNLIST

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo
and Howard Bilofsky and Harold Perry
Bolt Beranek and Newman Inc.

Summary

CONNLIST types out the connection table of a molecule.

Calling Sequence

```
RAC PUBLIC CONNLIST
CALL CONNLIST(mol-name)
              <molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

CONNLIST works on all types of terminals.

The public procedure CONNLIST types out the connection table of a molecule.

To use CONNLIST, type the command

```
CALL CONNLIST(mol-name)
```

where "mol-name" is the molecule whose connections you want listed.

For each atom in mol-name, CONNLIST lists its index, name (if it's named), and the names (or indices) of the atoms it is bonded to.

If desired, the public procedure MODIFY can produce a more complete connection table which also includes all bond types.

CRYST

by Michael Cory
Burroughs-Wellcome Co.

Summary

CRYST builds a PROPHET molecule from atom coordinates stored in a table.

Calling Sequence

```
RAC PUBLIC CRYST
CALL CRYST(atom-table)
             <table>
```

ENTRY

```
CALL CRYSTQT(atom-table, mol-name)
               <table>      <text>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

"Atom-table" must be in the format described below.

CRYST works on all terminals.

The public procedure CRYST uses a table containing atom coordinates to build a PROPHET molecule. The table containing the atomic coordinates must contain 5 columns:

Column 1: the <text> atom type
Column 2: the <text> atom name
Column 3: the <float> X-coordinate
Column 4: the <float> Y-coordinate
Column 5: the <float> Z-coordinate

To use CRYST, type the commands

```
CALL CRYST(atom-table)
```

Where "atom-table" is the name of the table containing the coordinates. CRYST calls the public procedure CONNECT through the MOLCONNECT entry, which in turn, calls %CONNECTMOL, to allow you to finish the construction of the molecule.

In CRYST, you will first be asked for the name of the constructed molecule. Next, you will be asked for a scale factor of the coordinates. Since X-ray coordinates are reported in many different scales, the correct scale factor needs to be found by trial and error. Usually 2 or 3 guesses should suffice .

If you don't desire any interaction with CRYST, you may invoke CRYST quietly via the entry

```
CRYSTQT(atom-table, mol-name)
```

where mol-name is the desired text name of the constructed molecule. CRYSTQT will use a scale factor of 1.0

See the following example for further illustration. The description of the public procedure CONNECT **in this chapter** and the procedure %CONNECTMOL in Chapter III, part 3, should be perused before using CRYST.

MOLDATA 25R X 5C

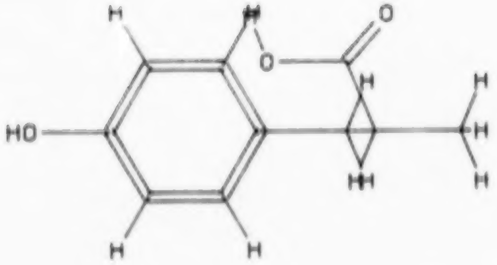
	1 ATOM TYPE	2 ATOM NAME	3 X	4 Y	5 Z
1	C	C1	4.964	0.	1.452
2	C	C2	3.424	0.	1.452
3	C	C3	2.91	0.	0.
4	C	C4	1.37	0.	0.
5	C	C5	2.91	1.257	2.177
6	O	O6	3.642	2.113	2.672
7	C	C7	.685	1.19	0.
8	C	C8	-.685	1.19	0.
9	C	C9	-1.37	0.	0.
10	C	C10	-.685	-1.19	0.
11	C	C11	.685	-1.19	0.
12	O	O12	-2.8	0.	0.
13	H	H13	-3.109	0.	.951
14	O	O14	1.48	1.257	2.177
15	H	H15	1.171	2.081	2.653
16	H	H16	5.331	0.	2.489
17	H	H17	5.331	.898	.933
18	H	H18	5.331	-.898	.933

MOLDATA 25R X 5C

	1 ATOM TYPE	2 ATOM NAME	3 X	4 Y	5 Z
19	H	H19	3.057	-.898	1.97
20	H	H20	3.276	-.898	-.519
21	H	H21	3.276	.898	-.519
22	H	H22	1.234	2.143	0.
23	H	H23	-1.234	2.143	0.
24	H	H24	-1.234	-2.143	0.
25	H	H25	1.234	-2.143	0.

(table supplied to CRYST)

```
>CRYST(MOLDATA)<
TYPE NAME OF CONSTRUCTED MOLECULE
 ?MOLQ< TYPE IN A SCALE FACTOR FOR SCALING THE COORDINATES
OF THE MOLECULE. IF NO CHANGE IS DESIRED TYPE 1.0 ?1.0<

                <INTERACTION WITH  CONNECT>


MOLECULE NAME IS MOLQ.
CONNECT COMPLETED. TOOK 41.83301 CPU SEC.

CRYST COMPLETED. TOOK 46.35699 CPU SEC.
```



constructed MOLQ displayed THREED COMPLETE

DISTAB

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo

Summary

DISTAB creates a table containing specified distances, angles, and/or torsion angles for molecules(s).

Calling Sequence

```
RAC PUBLIC DISTAB
CALL DISTAB
```

Citation

Potter, S. and Weeks, C.; "DISTAB" in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Gail Rubin on the PROPHET line.

Things to Remember

Models of the molecules must exist before calling DISTAB.

If you are using molecules which do not have named atoms, you must give atom indices.

DISTAB works on all types of terminals.

The public procedure DISTAB is designed to create a table containing specified distances, angles, and/or torsion angles for atoms of a molecule or molecules.

The molecules may be individual molecules, or they may be stored in a table or in an array.

To use DISTAB, type the command

CALL DISTAB

where you are first asked for the name of the variable containing the molecule(s). If it is a PROPHET molecule, type its name, if it is a table containing molecules, give the table's name. More than one table can be specified. If the input molecules are in table(s), you have a further option of performing the calculations on all molecules in the table(s), or on specified molecules.

You will be asked several questions about the calculation to be performed. You will be asked:

(a) if each molecule should be in a row or column. DISTAB is usually more efficient if you put each molecule in a column.

(b) if you want means and standard deviations computed (if you are doing more than one molecule)

(c) if the geometrical parameters are to be calculated for bonded atoms only

(d) if there are any equivalences between atoms (if you are doing more than one molecule)

(e) if distances, angles, and torsion angles are specified in the table

After answering these questions, begin entering the names or indices of the atoms to be considered, terminating by typing "NO" when asked for a name. If you wish to have information calculated for all atoms, answer "ALL". Note that if you say "ALL" for distance calculations, and you ansered "NO" to (b), you will get the distances for every pair of atoms in the molecule.

You will also be asked to supply a name for the table which will contain the informationcomputed by DISTAB.

When the message "DISTAB COMPLETE" is typed, you can display the output table which now contains the calculations you have specified.

See the following example for further elaboration.

MYMOLS 5R X 4C

0 NAME	1 TYPE	2 VALUE	3 MAKER	4 DATE
1. PR103	MOLECULE	$C_{23}\ H_{38}\ O_2$	MFB	10/17/79
2. PR104	MOLECULE	$C_{22}\ H_{35}\ O_2$	MFB	10/17/79
3. PR113	MOLECULE	$C_{23}\ H_{38}\ N_2$	MFB	10/17/79
4. PR115	MOLECULE	$C_{28}\ H_{35}\ F\ O_6$	MFB	10/17/79
5. PR116	MOLECULE	$C_{22}\ H_{36}\ O_2$	MFB	10/17/79

```
>DISTAB<
TYPE NAME OF VARIABLE CONTAINING MOLECULES?MYMOLS<
IF ADDITIONAL MOLECULE TABLES ARE TO BE SPECIFIED, TYPE TABLE NAMES.
END LIST WITH "NONE". TABLE NAME = ?NONE<
ARE ALL MOLECULES TO BE INCLUDED IN THE CALCULATIONS?N<
    TYPE THE NAMES OF THE MOLCULES TO BE CONSIDERED.
    TYPE "NONE" TO TERMINATE.
        MOLECULE NAME = ?PR103<
        MOLECULE NAME = ?PR104<
        MOLECULE NAME = ?PR113<
        MOLECULE NAME = ?NONE<
COLUMN CONTAINING MOLECULES ?2<

TYPE "C" OR "R" TO INDICATE WHETHER MOLECULE REGISTRY NUMBERS
ARE TO BE USED AS COLUMN HEADINGS OR ROW NAMES?C<

ARE MEANS AND STANDARD DEVIATIONS TO BE CALCULATED?Y<

ARE GEOMETRICAL PARAMETERS TO BE CALCULATED ONLY FOR BONDED ATOMS?N<

ARE THERE ANY EQUIVALENCES BETWEEN ATOM NAMES?N<

ARE DISTANCES, ANGLES, AND TORSIONAL ANGLES SPECIFIED IN OLD TABLE?N<

TYPE NAMES OR INDICES OF ATOMS FOR DISTANCE CALCULATION.
TYPE "ALL" TO DO ALL ATOMS. TYPE "NO" TO TERMINATE.
    NAME 1 = ?C5< NAME 2 = ?C6<
    NAME 1 = ?14< NAME 2 = ?15<
    NAME 1 = ?18< NAME 2 = ?C20<
    NAME 1 = ?NO<

TYPE NAMES OR INDICES OF ATOMS FOR ANGLE CALCULATION.
TYPE "ALL" TO DO ALL ATOMS,  TYPE "NO" TO TERMINATE.
    NAME 1 = ?C6<  NAME 2 = ?C7<  NAME 3 = ?C8<
    NAME 1 = ?20<  NAME 2 = ?21<  NAME 3 = ?22<
    NAME 1 = ?NO<

TYPE NAMES OR INDICES OF ATOMS FOR TORSION ANGLE CALCULATION.
TYPE "ALL" TO DO ALL ATOMS,  TYPE "NO" TO TERMINATE.
    NAME 1 = ?C1<  NAME 2 = ?C2<  NAME 3 = ?C3<  NAME 4 = ?C4<
    NAME 1 = ?C17<  NAME 2 = ?C18<  NAME 3 = ?C19<  NAME 4 = ?C20<
    NAME 1 = ?NO<

WHAT NAME WOULD YOU LIKE FOR THE OUTPUT TABLE?MOLINFO<
DISTAB COMPLETED. TOOK 13.558 CPU SEC.
```

MOLINFO 7R X 5C

0 ROWNAME	1 PR103	2 PR104	3 PR113	4 MEAN	5 STDEV
1. C5-C6	1.322	1.327	1.334	1.327667	.006027711
2. 14-15	1.529	1.529	1.537	1.531667	.004618803
3. 18-C20	3.177	3.197	3.007	3.127	.1044031
4. C6-C7-C8	112.5	114.1	113.3	113.3	.7999997
5. 20-21-22	35.	35.2	53.7	41.3	10.73918
6. C1-C2-C3-C4	57.7	57.9	57.5	57.7	.1999998
7. C17-C18-C19-C20	-25.6	-25.3	-27.8	-26.23333	1.36504

(output table produced by DISTAB)

EDITMODEL

by Douglas C. Rohrer
Medical Foundation of Buffalo
and
Mark Lavin
Bolt Beranek and Newman Inc.

Summary

EDITMODEL is used to modify the three-dimensional model of a molecule by manipulating atoms and bonds, adding atoms and fragment molecules at specified locations and orientations, and invoking an energy minimization procedure to refine three-dimensional coordinates.

Calling Sequence

```
RAC PUBLIC EDITMODEL <GO>
CALL EDITMODEL(MOLQ)
          <molecule>
```

Citation

Rohrer, D.; "EDITMODEL" in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Send message to 'ROHRER' using PROPHET.

Things to Remember

The molecule to be manipulated may have either 3-D model or 2-D sketch coordinates. In the latter case, 3-D coordinates are generated assuming z-values of 0.0, and the molecule must be REFINEd to achieve a reasonable 3-D structure.

EDITMODEL requires a display terminal.

EDITMODEL Is an interactive, graphic quiz similar to the one for MAKE MOLECULE (see Chapter II). The paragraphs below describe each of the commands in the EDITMODEL quiz, which may be invoked by pointing to the command "light buttons" (see Figure 1) with the data tablet pen, or typing the name of the command (commands can be abbreviated to any unambiguous prefix, e.g., REF for REFINE). Each of the command descriptions discusses the purpose of the command, specifies additional information which must be supplied (if applicable), any may include special notes or warnings.

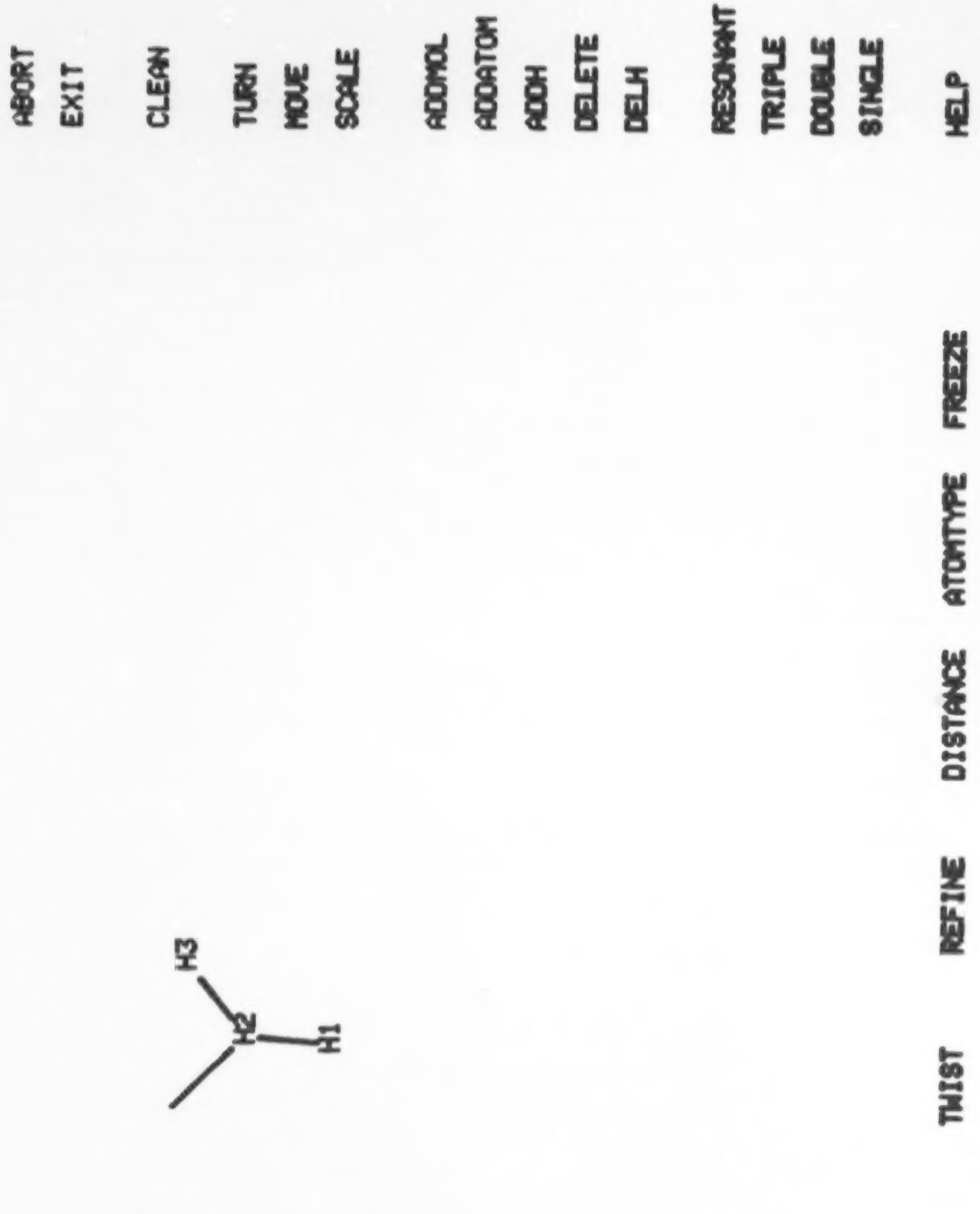

ABORT
EXIT
CLEAN
TURN
MOVE
SCALE
ADDMOL
ADDATOM
ADDH
DELETE
DELH
RESONANT
TRIPLE
DOUBLE
SINGLE
HELP
TWIST
REFINE
DISTANCE
ATOMTYPE
FREEZE
H3
H2
H1

FIGURE 1

COMMANDS

ABORT: Exit from EDITMODEL without incorporating any of the modifications made during the quiz; the molecule is restored to its pre-quiz state.

INPUT: EDITMODEL asks "DO YOU REALLY WANT TO ABORT (MOLECULE WILL BE RESTORED TO PRE-EDITMODEL STATE)?"; answer YES or NO.

NOTE: EDITMODEL preserves a copy of the molecule under the name BACKUPMOL, or BACKUPMOL1, or BACKMOL2, etc., in your private saved variables table. Normally the backup copy is deleted upon ABORTing or EXITing. In the event of an ABORT due to an error or system malfunction, you may have to remove the backup copy by using the DELETE SAVED command.

EXIT: Exit from the EDITMODEL quiz, incorporating the modifications made during the quiz.

INPUT: none

CLEAN: Refresh the display, removing unwanted commands and symbols and displaying the molecule in its current state.

INPUT: none

TURN: Change the display orientation of the molecule.

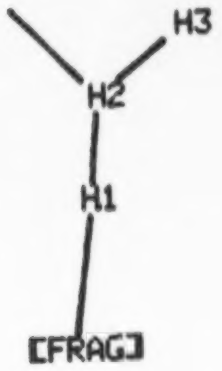
INPUT: Type in number of degrees to rotate the molecule (defaults to 90. if not specified) and a coordinate axis (X, Y, or Z) about which to rotate (defaults to Y), followed by <GO>.

NOTE: The FREEZE command must be used to preserve this new orientation upon EXITing.

MOVE:	Change the location of the molecule on the screen. This command is useful for making room in which to add atoms or fragments. INPUT: Point to a point on the screen, then to a new location for that point, and the molecule will be displaced by that amount.
SCALE:	Change the displayed size of the molecule. This command is useful in resolving overlapping atoms in a view by expanding the size of the molecule, or allowing room for more atoms by shrinking the size. INPUT: Type a scale factor (defaults to 0.5) and point to a point about which to scale or type <GO>. NOTE: The FREEZE command must be used to preserve this new scale upon EXITing.
ADDMOL:	To add a fragment molecule as a single unit, specifying a connection to the existing molecule and a location and orientation for the fragment unit. INPUT: EDITMODEL types "ENTER NAME OF FRAGMENT MOLECULE TO BE ATTACHED" - enter the name of the molecule in your work area to be attached. It must have 3-D MODEL coordinates. "DO YOU WANT TO DELETE AN ATOM FROM HOST MOLECULE?" - answer YES or NO (this is to make room for the connection of the fragment molecule to the host molecule). If YES, "POINT TO ATOM TO DELETE" - do so.

"POINT TO JOINING ATOM IN HOST MOLECULE" - point to an atom (call it "H1") to which the fragment molecule is to be attached.

A new atom (labeled "[FRAG]") and bond will appear, representing the link to the fragment being added see (Figure 2).



"POINT TO ATOM DEFINING BOND ANGLE" - point to an atom (call it "H2") connected to H1.

"ENTER ANGLE (DEGREES) FORMED BY <[FRAG], H1, H2 flash> " - type in the bond angle [FRAG]-H1-H2.

"POINT TO 4TH ATOM DEFINING TORSION ANGLE" - point to an atom (call it H3) connected to H2.

"ENTER TORSION ANGLE (DEGREES) FORMED BY <[FRAG], H1, H2 and H3 flash>" - enter the torsion angle [FRAG]-H1-H2-H3 in degrees. Remember that this may define the stereo-chemistry of H1, so drawing a rough sketch for yourself may help.

The fragment molecule is then displayed, and a similar series of questions is asked:

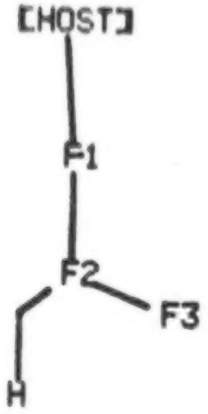
"DO YOU WANT TO DELETE AN ATOM FROM [FRAG] MOLECULE?" - answer YES or NO.

If YES, "POINT TO ATOM TO DELETE" - do so.

"POINT TO JOINING ATOM IN FRAG MOLECULE" - point to the atom to be connected to the host molecule (call this "F1").

A new atom labeled "[HOST]" and bond will appear, representing the link to the host molecule (see Figure 3).

"POINT TO ATOM DEFINING BOND ANGLE" - point to an atom (call it "F2") connected to F1.



"ENTER ANGLE (DEGREES) FORMED BY <[HOST], F1, F2 flash>" - type in the angle in degrees formed by F2-F1-[HOST].

"POINT TO 4TH ATOM DEFINING TORSION ANGLE" - point to an atom (call it F3) connected to F2.

"ENTER TORSION ANGLE (DEGREES) FORMED BY <[HOST], F1, F2, F3 flash>" - enter a torsion angle formed by [HOST]-F1-F2-F3. Remember the stereochemical implications.

The screen is cleared.

"WHAT TYPE OF BOND CONNECTS THE HOST AND FRAGMENT?" - enter S, D, T, or R for single, double, triple, or resonanat.

"DO YOU WANT TO CHANGE THE DEFAULT BOND LENGTH OF <length>?" - answer YES or NO.

If YES, "TO" - enter new bond length in angstroms.

The new, combined molecule is displayed.

"TO DEFINE TORSION ABOUT <HOST-FRAGMENT bond flashed> POINT TO AN ATOM ADJACENT TO <HOST atom flashed>" - do so.

"ENTER TORSION ANGLE (DEGREES) DEFINED BY <four atoms flash>" - enter the appropriate torsion angle for the specified bonds.

ADDATOM: Add a single atom to the molecule, specifying its type, bonding, and 3-D coordinates.

INPUT: EDITMODEL types "TYPE OF ATOM TO ADD" - type in atom type, e.g., C, H, O, etc.

"WHERE (ON DISPLAY SCREEN) SHOULD I STICK IT?" - point to an arbitrary location.

The new atom (call it "A1") is drawn at the indicated location for subsequent reference (of course, this is not its final position).

"POINT TO ATOM TO WHICH IT'S BONDED" - point to atom (call it "A2") in host molecule to be bonded to A1.

"WHAT IS THE TYPE OF BOND BETWEEN <A1 and A2 flash>" - type S, D, T, or R (single, double, triple, or resonant).

"CHANGE ASSUMED BOND LENGTH OF <length>?" - answer YES or NO.

If YES, "TO (ANGSTROMS)" - enter desired bond length.

"PICK A THIRD ATOM FOR BOND ANGLE" - point to an atom (call it "A3").

"WHAT SHOULD BE THE ANGLE <A1, A2, A3 flash>" - enter a valence (bond) angle for A1-A2-A3.

"PICK A FOURTH ATOM FOR TORSION ANGLE" - point to an atom to be bonded to A3 (call it "A4").

"WHAT SHOULD BE THE TORSION FOR <A1, A2, A3, and A4 flash>"? - enter the torsion angle. Remember that the torsion angle may also define the stereochemistry of the atoms.

ADDH: Adds hydrogen atoms (with 3-D coordinates) to balance valences at specified carbon atoms in the molecule.

INPUT: Point, one at a time, to the carbon atoms at which hydrogens are to be added,

OR

Type ALL to add hydrogens to all carbon atoms.

EDITMODEL will then type "ARE HYDROGEN NAMES TO BE SPECIFIED BY USER?" - if you answer YES, you will be prompted to supply names for all the hydrogens that are added.

NOTE: For ADDH to work correctly, the carbon atoms must already have the correct 3-D coordinates and the correct bonding to non-hydrogen atoms (since that would affect the number of hydrogens that must be added to achieve nominal valence).

DELETE: Delete atoms or bonds from the molecule.

INPUT: Point to the atoms or midpoints of the bonds to be deleted.

WARNING: Be careful not to delete the wrong atoms. TURN, SCALE or MOVE the display first to get a clear shot, if necessary.

DELH: Delete hydrogen atoms from the molecule.

INPUT: Point to a hydrogen to be deleted,
OR
Point to a non-hydrogen atom; all hydrogens attached to it will be deleted,
OR
Type ALL <GO> to remove all hydrogen atoms from the molecule.

RESONANT:
TRIPLE:
DOUBLE:
SINGLE: Add a bond between two atoms, or change the type of an existing bond.

INPUT: Point to two atoms; a bond of the specified type will be added between them, or, if they are already bonded, the bond will be

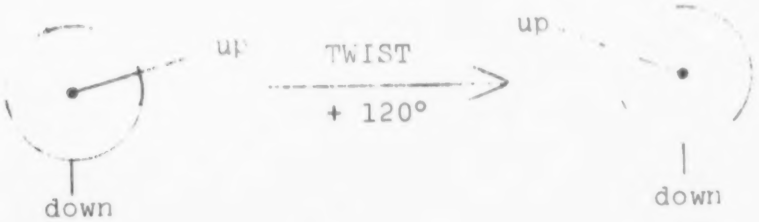
changed to the specified type,
OR
Point to an existing bond; it will be changed to the specified bond type.

WARNING: Changing the type of a bond will affect valence balance, but EDITMODEL will not check for or correct this. Thus, it may be desirable, for example, to use DELH, followed by SINGLE, DOUBLE, etc., followed by ADDH.

TWIST: Change the conformation of part of the molecule by twisting about a rotatable bond.

INPUT: Type in the number of degrees to TWIST, and then point to the two atoms at the ends of the bond to twist about. The part of the molecule connected to the second of the atoms will be rotated counterclockwise when viewed looking from the second atom toward the first atom, if the angle to TWIST was positive, as shown in Figure 4.

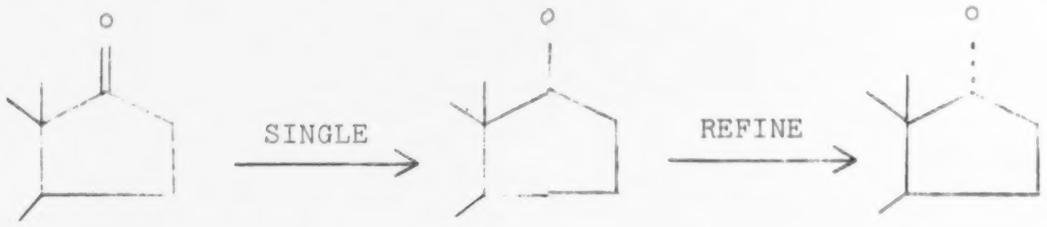
WARNING: You can only twist about a single bond. To change a CIS- to a TRANS-, it is necessary to convert the double bond using the SINGLE command, then TWIST, then convert back using the DOUBLE command.



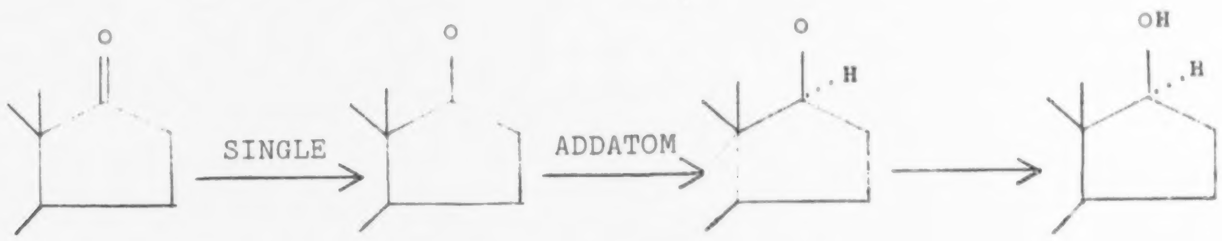
REFINE: To refine the geometry of the molecule using molecular mechanics. The molecule may be refined with or without including hydrogens. The actual refinement is done by a FORTRAN program outside of PROPHET. Currently, the only atoms that can be used are hydrogen, carbon, nitrogen and oxygen. If refinement of molecules containing other atom types is desired, special arrangements will have to be made. Even with the limited atom types, some unusual geometric types may not be covered within the program. If a "STOP" message occures, most likely the molecule has one of these unusual types. Under these circumstances, it is best to contact your EDITMODEL representative for help. After a "STOP" message, the refinement has not been successful and you are not in PROPHET. You may return to PROPHET by typing "CALL RUNRTN". Unfortunately, EDITMODEL will probably abort because the information needed to update the molecule after refinement has not been generated. The molecule still exists in your work area as it was before refinement was attempted. It is probably best to save it and seek help.

WARNING: Refinement of this type is reasonably costly and may also require a long wait to complete. It is not unusual for a 25 atom refinement to take 5 to 8 minutes of CPU time and an elapsed time of anywhere up to an hour. In addition, the time required increases as the square of the number of atoms. Thus, doubling the number of atoms will increase the CPU time by a factor of about 4. It may be best to do refinements in batch mode at night when it costs less and you will not have to sit and wait for the refinement to finish. The public procedure REFINEMOL can be used to set up a batch job, the molecule must be saved private.

NOTE: Experience has shown that for best results the hydrogen atoms should be included. However, if the molecule started as only a sketch, or if bond type or atom type changes have been made, the model can be improved greatly by doing a refinement without hydrogens or only those hydrogens absolutely needed. The hydrogen atoms can be added and deleted readily with the ADDH and DELH commands. Specific hydrogen atoms can be added using ADDATOM when the geometry is wrong but the stereochemistry must be specified. For example:



changing the double bond to a single bond, followed by refinement will result in an alpha oxygen position.



This same sequence, but with a hydrogen added on the alpha side using ADDATOM will result in beta oxygen position.

This same procedure can often be used to edit a molecule from one type to another and obtain the desired result.

DISTANCE: Types the distance between any two atoms in the molecule. This may be useful after the conformation or nature of the molecule has been changed using TWIST, REFINE, ADDATOM, ADDH, or ADDMOL.

INPUT: Point to a pair of atoms; the distance in angstroms between them will be typed,

OR

Point to the midpoint of a bond; its length, in angstroms, will be typed.

ATOMTYPE: Changes the type of an atom:

INPUT: EDITMODEL will prompt "ENTER TYPE OF ATOM" - type in a chemical symbol, e.g., H, C, O, etc. Then point to an atom or atoms; the type of each will be changed to the specified symbol.

NOTE: As elsewhere in PROPHET, carbon atoms will be drawn as unlabeled points; atoms are turned into carbons, though,by setting their type to C.

FREEZE: Preserves the effect of all SCALE and TURN commands upon exit from the EDITMODEL quiz.

INPUT: none.

HELP: Types a brief description of one or all EDITMODEL commands.

INPUT: Typing GO or ALL followed by <GO> causes EDITMODEL to clear the screen, print a list of all EDITMODEL commands with one line descriptions, and, when you are ready (signified by typing <GO>) restoring the picture of the molecule.

Alternatively, you can get slightly more detailed information about a single command by pointing to the appropriate light-button or typing the command name (or any unambiguous prefix).

References

The molecular mechanics refinement is done by an external FORTRAN program QCFF, originally written by E. Huber, R. Sharon, and A Warshel (1974). The following references provide some details about the QCFF program:

E. Huber and A. Warshel, Acta Crystallogr., B30, 1822 (1974).
A. Warshel and S. Lifson, J. Chem. Phys., 53, 582 (1970).
A. Warshel and M. Karplus, J. Am. Chem. Soc., 94, 5613 (1972).
S. Lifson and A. Warshel, J. Chem. Phys., 49, 5116 (1968).

FITMOL

by Douglas Rohrer
and G. David Smith
Medical Foundation of Buffalo

Summary

FITMOL fits a fragment molecule to a parent molecule by minimizing the distances between a user selected subset of atoms in each molecule.

Calling Sequence

```
RAC PUBLIC FITMOL
CALL FITMOL
```

Citation

Rohrer, D. and Smith, G.D.; "FITMOL" in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

FITMOL works on all terminals. Models for each molecule must have been computed.

FITMOL alters the display of the fragment molecule so as to minimize the distances between a set of atoms in the fragment and parent molecules. This is accomplished in part by iteratively computing a rotation transformer for the fragment molecule. The user can control the number of iterations used. The process converges quite fast and usually only 3 iterations are needed. Once the fragment is fitted to the parent, the user may request the distances between any atoms in the parent and fragment.

Prior to the fitting, the user is given the option of orienting the parent or fragment molecule. (FITMOL calls the public procedure ORMOL to do this).

The following example illustrates the use of FITMOL.

After specifying the parent molecule, the user must specify a set of atoms to which the fragment molecule could be fitted to. The same must then be done for the fragment molecule.

The fitted molecules may be displayed in their correct relation by forming them into a multimolecule, thus:

```
DECLARE  MM  MULTIMOLECULE,
         IM  FIXED;

CALL  %MAKEMULTIMOL(MM)
CALL  %MMADDMOL (MM, <molecule1>, IM) /* First molecule */
              .
              .
              .
CALL  %MMADDMOL (MM, <moleculeN>, IM) /* N-th molecule */
CALL  %ADJUSTMULTIMOL (MM)
DISPLAY  MM
```

```
>FITMOL<
TYPE NAME OF PARENT MOLECULE:?AN65<
SHOULD MOLECULE BE RE-ORIENTED?N<
TYPE NAMES OR SERIAL NUMBERS OF ATOMS TO BE FIT.
THE FORM IS: NAME1, NAME2 TO NAMEN, NAMEM
?C1 TO C9<
TYPE NAME OF MOLECULE TO BE FIT TO PARENT?AN78<
SHOULD MOLECULE BE RE-ORIENTED?N<
TYPE NAMES OR SERIAL NUMBERS OF ATOMS TO BE FIT
IF SAME AS PARENT, HIT <GO>
?<
TYPE CONVERGENCE LIMIT (RMS DIFFERENCE BETWEEN CYCLES, 0.01 IS REASONAB-
LE)?.01<

NUMBER OF ITERATIONS = 2
CHANGE IN AVERAGE SEPARATION LAST ITERATION = .001060924
AVERAGE INTERMOLECULE SEPARATION = .1170753
STANDARD DEVIATION OF THE AVERAGE = .06751946
SUM OF SQUARES OF THE SEPARATION = .1643896

LIST INDIVIDUAL INTERATOMIC SEPARATIONS?N<
DO ANOTHER SET OF ITERATIONS?Y<

NUMBER OF ITERATIONS = 3
CHANGE IN AVERAGE SEPARATION LAST ITERATION = .001061752
AVERAGE INTERMOLECULE SEPARATION = .1170745
STANDARD DEVIATION OF THE AVERAGE = .06752089
SUM OF SQUARES OF THE SEPARATION = .1643896

LIST INDIVIDUAL INTERATOMIC SEPARATIONS?Y<


PAR   FIT      D        PAR   FIT      D        PAR   FIT      D
C1    C1     0.233      C4    C4     0.042      C7    C7     0.088
C2    C2     0.044      C5    C5     0.043      C8    C8     0.141
C3    C3     0.215      C6    C6     0.131      C9    C9     0.117


DO ANOTHER SET OF ITERATIONS?N<
LIST ANY OTHER INTERATOMIC SEPARATIONS?Y<
TYPE NAMES OR SERIAL NUMBERS OF ATOMS IN PARENT.
?C14 TO C20<
TYPE NAMES OR SERIAL NUMBERS OF ATOMS IN FIT MOLECULE.
IF SAME AS PARENT, HIT <GO>
?C1 TO C7<


PAR   FIT      D        PAR   FIT      D        PAR   FIT      D
C14   C1     4.996      C17   C4     8.076      C20   C7     4.126
C15   C2     7.728      C18   C5     5.648
C16   C3     8.907      C19   C6     3.542


LIST ANY OTHER INTERATOMIC SEPARATIONS?N<
FIT ANOTHER MOLECULE TO THE PARENT MOLECULE?N<

FITMOL COMPLETED. TOOK 7.315998 CPU SEC.
```

GENMOL

by Douglas Rohrer
Medical Foundation of Buffalo

Summary

GENMOL alters the display of a fragment molecule so as to simulate the user controlled linking of the fragment to a parent molecule. It then produces a table of the imagespace coordinates of the linked molecules.

Calling Sequence

```
RAC PUBLIC GENMOL
CALL GENMOL
```

Citation

Rohrer, D.; "GENMOL", in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Send message to 'ROHRER' using PROPHET.

Things to Remember

Models must have been computed for the parent and fragment molecules. GENMOL works on all terminals.

In the following (rather ad hoc) example, the user begins by:

1) Specifying the parent and fragment molecules

2) Specifying those atoms that should not be included in the coordinates table
3) Giving a name and approximate size to the coordinates table.

The user is then asked for the linking information which involves specifying:

1) atoms X3,X2,X1 of the Parent
2) atoms Y3, Y2, Y1 of the Fragment
3) the bondlength between Y1 and X1

and 4) the various bond and torsion angles

Once completed, the table NEWMOLDATA contains the 3-D display coordinates of the simulated linking.

The linked molecules may be displayed in their linked relation by forming them into a multimolecule, thus:

```
DECLARE MM MULTIMOLECULE
CALL %MAKEMULTIMOL(MM)
CALL %MMADDMOL(MM, <molecule 1>)   /* first molecule */
    .
    .
    .
    .
CALL %MMADDMOL(MM, <molecule N>)   /* last molecule */
DISPLAY MM
```

```
>GENMOL<
INPUT THE NAME OF THE PARENT MOLECULE ?PMOL<
SHOULD ANY ATOMS BE DELETED?N<
INPUT THE NAME OF THE TABLE TO CONTAIN THE COORDINATES
OF THE NEW MOLECULE?NEWMOLDATA<
INPUT THE APPROXIMATE TOTAL NUMBER OF ATOMS?50<
INPUT THE NAME OF THE FRAGMENT MOLECULE TO BE LINKED TO THE
PARENT MOLECULE ?FMOL<
SHOULD ANY ATOMS BE REMOVED FROM THE FRAGMENT?Y<
INPUT THE NAMES OR SERIAL NUMBERS OF THE ATOMS TO BE REMOVED
?11 TO 13<                                      FRAGMENT

                                           Y2------Y3
                                          /
                                         /
    PARENT                              /
                                       /
                             X1......Y1
                            /
                           /
                          /
                         /
    X3------X2

USING THE BONDING SCHEME SHOWN ABOVE,
THE FOLLOWING INFORMATION IS REQUIRED TO LINK THE
FRAGMENT TO THE PARENT.


        INPUT THE NAMES OR SERIAL NUMBERS OF ATOMS
        X3, X2, AND X1 OF THE PARENT MOLECULE.
        ?6,7,10<
        INPUT THE NAMES OR SERIAL NUMBERS OF ATOMS
        Y1, Y2, AND Y3 OF THE FRAGMENT MOLECULE.
        ?4,3,2<
 INPUT THE LENGTH OF THE CONNECTING BOND X1....Y1?1.5<

 INPUT BOND ANGLE X2----X1....Y1?45<
 INPUT BOND ANGLE X1....Y1----Y2?45<
 INPUT THE FOLLOWING TORSION ANGLES:


        X3----X2----X1....Y1?0<
        X2----X1....Y1----Y2?0<
        X1....Y1----Y2----Y3?90<
 DO YOU WANT TO ADD ANOTHER FRAGMENT?N<

 GENMOL COMPLETE. TOOK 18.307 CPU SEC.
```

NEWMOLDATA 25R X 5C

	1 ATOM NAME	2 ATOM TYPE	3 X	4 Y	5 Z
1	1	H	0.	0.	0.
2	2	O	.96	0.	0.
3	3	C	1.413977	1.281992	0.
4	4	O	.6216507	2.209688	0.
5	5	C	2.849529	1.548056	0.
6	6	C	3.670819	2.743039	0.
7	7	C	4.865802	1.92175	0.
8	8	C	4.044512	.7267672	0.
9	9	H	3.474004	3.804954	0.
10	10	H	5.927717	2.118564	0.
11	11	N	4.299642	-.6497897	0.
12	12	H	3.433617	-1.14979	0.
13	13	H	4.82489	-.8919324	.8157704
14	14	H	4.82489	-.8919324	-.8157704
15	1	H	4.924157	1.71302	1.907784
16	2	O	5.037775	1.09999	1.177795
17	3	C	4.913855	1.768603	0.
18	4	O	4.691528	2.968174	0.

NEWMOLDATA 25R X 5C

	1 ATOM NAME	2 ATOM TYPE	3 X	4 Y	5 Z
19	5	C	5.046887	1.050827	-1.264397
20	6	C	4.978496	1.41983	-2.66499
21	7	C	5.233734	.04269068	-3.040277
22	8	C	5.302124	-.3263127	-1.639685
23	9	H	4.80805	2.339476	-3.20499
24	10	H	5.332141	-.488267	-3.975584
25	14	H	4.816682	-2.183561	-1.181827

(table of coordinates produced by GENMOL. Note that atoms 11-13 are missing from the fragment molecule.)

MAKEPOLYPEPTIDE

by Wayne Rindone

Bolt Beranek and Newman Inc.

Summary

MAKEPOLYPEPTIDE creates a molecule containing a specified sequence of amino acids with a conformation specified by a series of phi and psi torsion angles in the polypeptide backbone.

Calling Sequence

```
RAC PUBLIC MAKEPOLYPEPTIDE

DECLARE unused-name MOLECULE

CALL MAKEPOLYPEPTIDE(unused-name)
                     <molecule>
```

Entry

```
CALL MAKEPOLYPEPTIDEQT(residue-list, phi-angles,
                       <text array> <float array>

            psi-angles,  mol-names, unused-name)
           <float array><text array><molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Wayne Rindone on the PROPHET line.

Things to Remember

MAKEPOLYPEPTIDE works on all terminals.

MAKEPOLYPEPTIDE creates molecules containing three-dimensional models; it is not necessary to use the COMPUTE MODEL command. The procedure does, however, set up a constraints table in the molecule much like that created by the public procedure SETPEPTIDEBONDS, so that it is possible for you to make modifications in the molecule and then use the COMPUTE MODEL command without changing the conformation of those parts of the molecule that you have not modified.

best copy available

The public procedure MAKEPOLYPEPTIDE automatically creates polypeptide molecules from a collecton of amino acid fragments stored in the public table AMINOACIDS. Certain columns of public AMINOACIDS, as it exists at the present time, are illustrated in Figure 1.

AMINOACIDS 22R X 8C

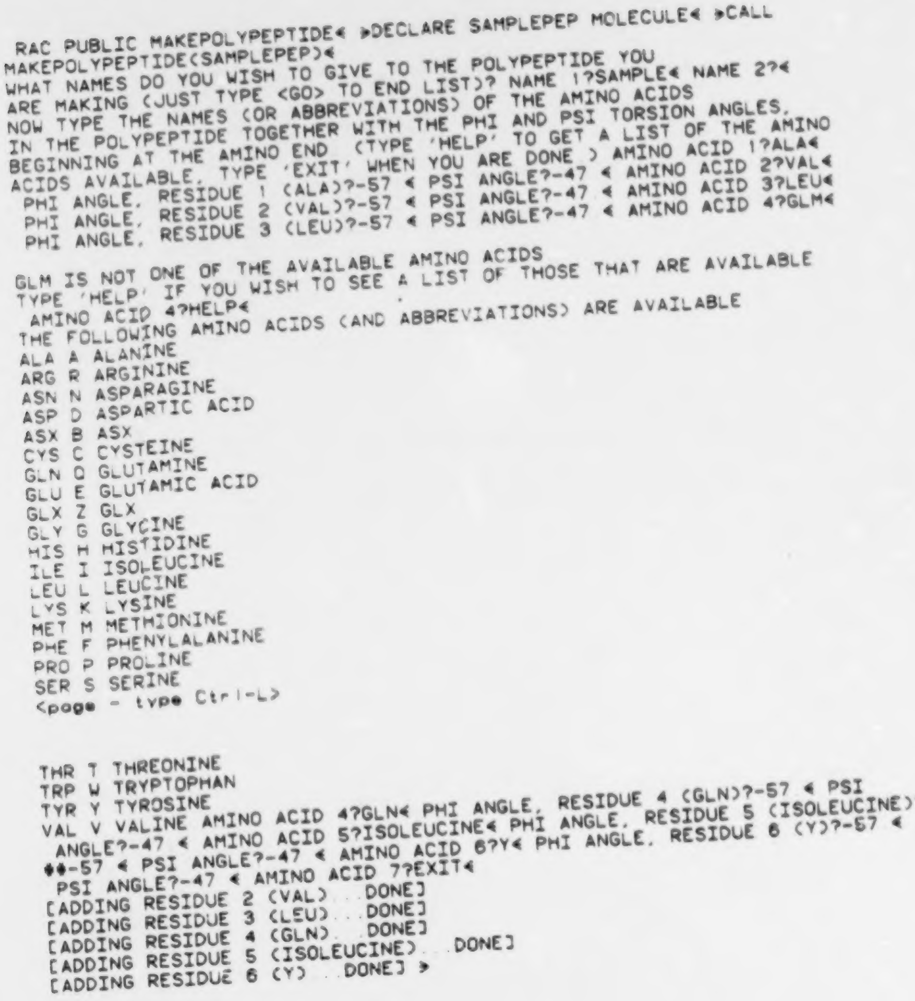
0 NAME	1 ENTRY	7 ABB 1	8 ABB 2
1 ALANINE	$C_3 H_5 N O ?_2$	ALA	A
2 ARGININE	$C_6 H_{14} N_4 O ?_2$	ARG	R
3 ASPARAGINE	$C_4 H_6 N_2 O_2 ?_2$	ASN	N
4 ASPARTIC ACID	$C_4 H_5 N O_3 ?_2$	ASP	D
5 ASX	$C_4 H_4 A N O_2 ?_2$	ASX	B
6 CYSTEINE	$C_3 H_5 N O S ?_2$	CYS	C
7 GLUTAMINE	$C_5 H_8 N_2 O_2 ?_2$	GLN	Q
8 GLUTAMIC ACID	$C_5 H_7 N O_3 ?_2$	GLU	E
9 GLX	$C_5 H_6 A N O_2 ?_2$	GLX	Z
10 GLYCINE	$C_2 H_3 N O ?_2$	GLY	G
11 HISTIDINE	$C_6 H_7 N_3 O ?_2$	HIS	H
12 ISOLEUCINE	$C_6 H_{11} N O ?_2$	ILE	I
13 LEUCINE	$C_6 H_{11} N O ?_2$	LEU	L
14 LYSINE	$C_6 H_{13} N N* O ?_2$	LYS	K
15 METHIONINE	$C_5 H_9 N O S ?_2$	MET	M
16 PHENYLALANINE	$C_9 H_9 N O ?_2$	PHE	F
17 PROLINE	$C_5 H_7 N O ?_2$	PRO	P
18 SERINE	$C_3 H_5 N O_2 ?_2$	SER	S
19 THREONINE	$C_4 H_7 N O_2 ?_2$	THR	T
20 TRYPTOPHAN	$C_{11} H_{10} N_2 O ?_2$	TRP	W
21 TYROSINE	$C_9 H_9 N O_2 ?_2$	TYR	Y
22 VALINE	$C_5 H_9 N O ?_2$	VAL	V

Figure 1. Public AMINOACIDS table.

Molecules created with MAKEPOLYPEPTIDE and its quiet entry MAKEPOLYPEPTIDEQT have the following features:

1. Two-dimensional sketch coordinates so that you can EDIT the molecule.
2. Three-dimensional model coordinates.
3. Atom labels ('LABEL') on the beta carbon atoms in each residue so that you can display the molecule with the DISPLAY ... LABELING ATOMS ... command.
4. A selector column ('BACKBONE') in the ATOMS table that marks each atom that belongs to the polypeptide backbone so that you can selectively display only the backbone atoms.
5. A CONSTRAINTS table containing all the appropriate constraints about each peptide bond so that you can recreate the model of the molecule by using the COMPUTE MODEL command.

Figure 2 illustrates the use of MAKEPOLYPEPTIDE to create SAMPLEPEP, a small sample polypeptide with the conformation of a right-handed alpha helix. The amino acids in the polypeptide can be specified either by their full common names, by standard three-letter codes or by standard one-letter codes.



```
RAC PUBLIC MAKEPOLYPEPTIDE◂ ▸DECLARE SAMPLEPEP MOLECULE◂ ▸CALL
MAKEPOLYPEPTIDE(SAMPLEPEP)◂
WHAT NAMES DO YOU WISH TO GIVE TO THE POLYPEPTIDE YOU
ARE MAKING (JUST TYPE <GO> TO END LIST)? NAME 1?SAMPLE◂ NAME 2?◂
NOW TYPE THE NAMES (OR ABBREVIATIONS) OF THE AMINO ACIDS
IN THE POLYPEPTIDE TOGETHER WITH THE PHI AND PSI TORSION ANGLES,
BEGINNING AT THE AMINO END  (TYPE 'HELP' TO GET A LIST OF THE AMINO
ACIDS AVAILABLE. TYPE 'EXIT' WHEN YOU ARE DONE ) AMINO ACID 1?ALA◂
 PHI ANGLE, RESIDUE 1 (ALA)?-57 ◂ PSI ANGLE?-47 ◂ AMINO ACID 2?VAL◂
 PHI ANGLE, RESIDUE 2 (VAL)?-57 ◂ PSI ANGLE?-47 ◂ AMINO ACID 3?LEU◂
 PHI ANGLE, RESIDUE 3 (LEU)?-57 ◂ PSI ANGLE?-47 ◂ AMINO ACID 4?GLM◂

GLM IS NOT ONE OF THE AVAILABLE AMINO ACIDS
TYPE 'HELP' IF YOU WISH TO SEE A LIST OF THOSE THAT ARE AVAILABLE
 AMINO ACID 4?HELP◂
THE FOLLOWING AMINO ACIDS (AND ABBREVIATIONS) ARE AVAILABLE
ALA A ALANINE
ARG R ARGININE
ASN N ASPARAGINE
ASP D ASPARTIC ACID
ASX B ASX
CYS C CYSTEINE
GLN Q GLUTAMINE
GLU E GLUTAMIC ACID
GLX Z GLX
GLY G GLYCINE
HIS H HISTIDINE
ILE I ISOLEUCINE
LEU L LEUCINE
LYS K LYSINE
MET M METHIONINE
PHE F PHENYLALANINE
PRO P PROLINE
SER S SERINE
<page - type Ctrl-L>


THR T THREONINE
TRP W TRYPTOPHAN
TYR Y TYROSINE
VAL V VALINE AMINO ACID 4?GLN◂ PHI ANGLE, RESIDUE 4 (GLN)?-57 ◂ PSI
 ANGLE?-47 ◂ AMINO ACID 5?ISOLEUCINE◂ PHI ANGLE, RESIDUE 5 (ISOLEUCINE)?
◆◆-57 ◂ PSI ANGLE?-47 ◂ AMINO ACID 6?Y◂ PHI ANGLE, RESIDUE 6 (Y)?-57 ◂
 PSI ANGLE?-47 ◂ AMINO ACID 7?EXIT◂
[ADDING RESIDUE 2 (VAL)...DONE]
[ADDING RESIDUE 3 (LEU)...DONE]
[ADDING RESIDUE 4 (GLN)...DONE]
[ADDING RESIDUE 5 (ISOLEUCINE)...DONE]
[ADDING RESIDUE 6 (Y)...DONE] ▸
```

Figure 2. The use of MAKEPOLYPEPTIDE to create SAMPLEPEP.

Figures 3 to 5 show several different ways in which the molecule SAMPLEPEP created by interactions in Figure 2 can be displayed. Figure 6 shows the first few rows of the ATOMS table of SAMPLEPEP.

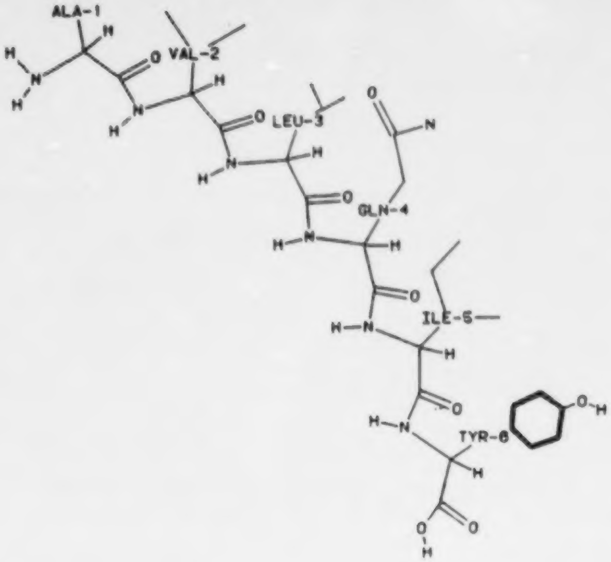

ALA-1
VAL-2
LEU-3
GLN-4
ILE-5
TYR-6

Figure 3. Display created by commands

```
EAD SAMPLEPEP LABELING ATOMS 'LABEL' <GO>
SCALE <GO>
```

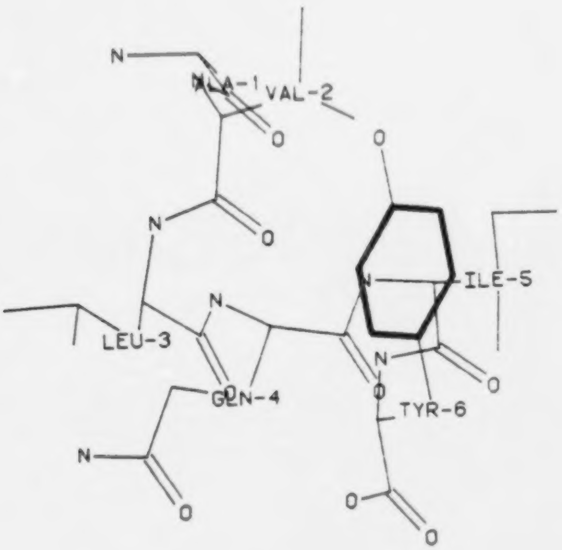

ALA-1
VAL-2
LEU-3
GLN-4
ILE-5
TYR-6

Figure 4. Display created by command

```
EAD SAMPLEPEP THREED LABELING ATOMS 'LABEL' <GO>
```

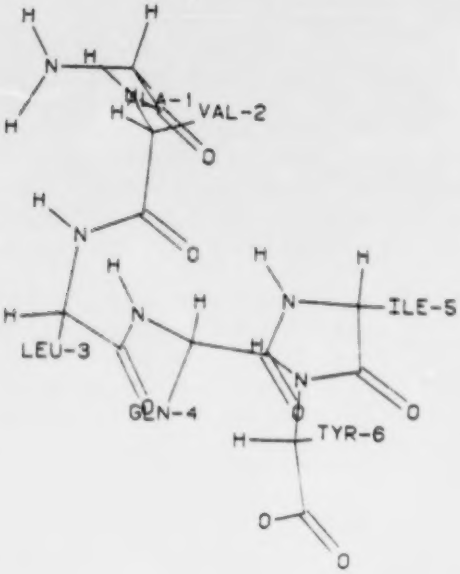


Figure 5. Display created by the command

EAD SAMPLEPEP THREED SELECTING ATOMS 'BACKBONE'
LABELING ATOMS 'LABEL' <GO>

ATOMS 105R X 9C

0 ATOM NAMES	1 TYPE	2 BOND ARRAY	3 SKETCH X	4 SKETCH Y	5 MODEL X	6 MODEL Y	7 MODEL Z	8 BACKBONE	9 LABEL
1. ALA 1 N	N	[1,6,7]	3.86	-6.00	0.00	0.00	0.00	TRUE	
2. ALA 1 CA	C	[1,2,4,5]	5.70	-5.00	1.45	0.00	0.02	TRUE	
3. ALA 1 C	C	[2,3,26]	7.16	-6.86	2.00	-.77	1.18	TRUE	
4. ALA 1 O	O	[3]	8.54	-5.20	2.00	-1.60	1.03	TRUE	
5. ALA 1 HA	H	[5]	6.72	-4.20	1.79	1.04	.89	TRUE	
6. ALA 1 HN	H	[6]	3.24	-7.06	-.45	.92	0.00	TRUE	
7. ALA 1 HN2	H	[7]	2.00	-5.04	-.70	-1.12	0.00	TRUE	
8. ALA 1 CB	C	[4,8,9,10]	5.54	-3.50	2.01	-.56	-1.32	TRUE	ALA-1
9. ALA 1 HB1	H	[8]			1.80	.81	-2.17		
10. ALA 1 HB2	H	[9]			1.71	-1.62	-1.42		
11. ALA 1 HB3	H	[10]			3.10	-.48	-1.33		
12. VAL 2 N	N	[11,16,20]	7.88	-7.30	1.45	-.49	2.38	TRUE	
13. VAL 2 CA	C	[11,12,14,15]	9.90	-6.66	1.87	-1.15	3.58	TRUE	
14. VAL 2 C	C	[12,13,45]	11.00	-8.01	1.73	-2.05	3.40	TRUE	
15. VAL 2 O	O	[13]	12.52	-7.56	2.60	-3.40	3.81	TRUE	
16. VAL 2 HA	H	[15]	10.90	-6.15	1.20	-.62	4.39	TRUE	
17. VAL 2 HN	H	[16]	8.95	-8.80	.70	.29	2.45	TRUE	
18. VAL 2 CB	C	[14,17,18,19]	10.80	-5.14	3.31	-.78	3.96	TRUE	VAL-2
19. VAL 2 HB1	H	[17]			3.58	-1.22	4.02		
20. VAL 2 CG1	C	[18,23,24,25]	11.90	-4.85	3.41	.77	4.07		
21. VAL 2 CG2	C	[19,20,21,22]	8.92	-3.54	4.28	-1.25	2.88		
22. VAL 2 HG1	H	[20]			5.31	-.07	3.13		
23. VAL 2 HG2	H	[21]			4.81	-.88	1.90		
24. VAL 2 HG3	H	[22]			4.21	-2.35	2.78		
25. VAL 2 HG1	H	[23]			4.44	1.06	4.34		
26. VAL 2 HG2	H	[24]			2.72	1.13	4.85		
27. VAL 2 HG3	H	[25]			3.14	1.23	3.11		
28. LEU 3 N	N	[27,32,45]	11.41	-9.37	.60	-3.80	2.96	TRUE	
29. LEU 3 CA	C	[27,28,30,31]	13.52	-9.21	.32	-4.51	2.79	TRUE	

Figure 6. First 29 rows of ATOMS of SAMPLEPEP.

NOTES

1. The COMPUTE MODEL command in PROPHET does not take constraints into account when they pertain to atoms and bonds in ring structures. If you EDIT a polypeptide created by MAKEPOLYPEPTIDE to close a large ring (such as by adding a disulfide bridge) and then use the COMPUTE MODEL command, all of the constraints MAKEPOLYPEPTIDE previously set for such things as the phi and psi torsion angles will be ignored; the results are thus not likely to be very useful.

2. You can set "omega" torsion angles (the torsion angles about the peptide bond itself) to values other then 180 degrees. To do so, modify the 180 degree value for the torsion angle constraint about the peptide bond in question and then use the COMPUTE MODEL command. The constraint value may be modified by using the CONSTRAIN command (see Chapter II, part 2), the system procedure %SETCONVALUE (see Chapter II, part 3) or the system procedure %SETCONTORSION (see Chapter III, part 3). Alternatively, you may use the TWIST command (see Chapter II, part 3), or the system procedure %TWISTMOL (see Chapter III, part 3). Note, however, that any twists you do will not be preserved during a subsequent COMPUTE MODEL command unless you also set an appropriate torsion angle constraint.

3. Similarly, you can set torsion angles in the amino acid side chains either by setting appropriate constraints or by twisting the side chain in question.

4. The fragments stored in the public table AMINOACIDS are all L-amino acids. If any of the residues in the polypeptide you wish to create are D-amino acids, use MAKEPOLYPEPTIDE to initially create the molecule containing only L-amino acids, and then use the system procedure %SETCONBONDS to modify the order in which the bonds are mentioned in the bond angles constraint about the alpha carbon, as described in the notes at the end of the descripton of the public procedure SETPEPTIDEBONDS.

5. If you have a private AMINOACIDS table or have accessed a sharable AMINOACIDS table, your table will be used by MAKEPOLYPEPTIDE instead of the public AMINOACIDS table. The following addendum provides details of how to utilize your own version of the AMINOACIDS table. You are welcome to propose any amino acids that you believe are common enough to add to those kept in the public AMINOACIDS table.

6. MAKEPOLYPEPTIDE consumes a significant amount of computer time (3 to 9 seconds per amino acid, depending on the length of the polypeptide and the size of the side chains). We encourage you to consider running MAKEPOLYPEPTIDEQT from a batch job whenever you are creating a molecule containing more than a few amino acids. See the description of %AUTOBAT in the Public Procedures Notebook.

7. Any polypeptides containing ten or more amino acids that MAKEPOLYPEPTIDE creates are set up as disk-based molecules. See Chapter II, part 5, for details about using such molecules.

ADDENDUM

Creating Your Own Amino Acid Fragments

It is important, if you set up your own AMINOACIDS table containing amino acid fragments not found in the public table, that you follow the conventions that MAKEPOLYPEPTIDE relies on to perform its function. Some of these conventions are as follows:

1. The private or sharable table in question must be named AMINOACIDS and must have the same structure and column headings as the public AMINOACIDS table.

2. In addition to any special amino acid fragments that you have in the table, it should also contain the 22 common amino acid fragments contained in the public table.

3. It is imperative that certain of the atoms in each fragment have the atom names that MAKEPOLYPEPTIDE looks for, as follows:

   the carbonyl carbon atom must be named 'C'

   the carbonyl oxygen atom must be named 'O'

   the amino nitrogen atom must be named 'N'

   the hydrogen atom (if there is one) on the amino nitrogen atom must be named 'HN'

   the alpha carbon atom must be named 'CA'

   the beta carbon atom (if there is one) must be named 'CB'

   there must be an atom of type '?' attached by a single bond to the amino nitrogen and named 'NEXIT'

   there must be an atom of type '?' attached by a single bond to the carbonyl carbon atom and named 'CEXIT'

4. Each amino acid fragment must have a three-dimensional model and should have constraints set that will cause the same model to be regenerated if the COMPUTE MODEL command is used.

5. The atom of each fragment that will belong to the polypeptide backbone should be marked (by the MARK ATOMS command, Chapter II, part 3, or by the system procedure %MARKATOMS, Chapter III, part 3) as belonging to 'BACKBONE'

6. There should be a TEXT column in the ATOMS table of each fragment with the column heading 'LABEL' containing, for one of the atoms in the fragment (preferably the beta cabon) the label you wish to have displayed for that amino acid.

The easiest way to make sure you follow all these conventions is to use the public AMINOACIDS table and some amino acid fragment from that table as templates. For example, suppose you wish to have a private AMINOACIDS table containing pyroglutamic acid in addition to the other more common amino acids. You could do this with the following commands:

```
ACCESS PUBLIC AMINOACIDS <GO>
MAKE TABLE MYAA FROM AMINOACIDS <GO>
CLEAR AMINOACIDS <GO>
RENAME MYAA TO AMINOACIDS <GO>
SET GLU TO ROW 'GLUTAMIC ACID' COL 1
       OF AMINO ACIDS <GO>
MAKE MOLECULE PYROGLU FROM GLU <GO>


  .
  .
  .

. at this point you enter the MAKE MOLECULE
. quiz where you can remove a water molecule
. from the glutamic acid molecule and join
. the amino nitrogen to the gamma carbon to
. make pyroglutamic acid

COMPUTE MODEL OF PYROGLU <GO>

. at this point you enter the WIPKE quiz,
. after which PROPHET carries out an energy
. minimization to compute a model for the
. ring structure in pyroglutamic acid

ADD ROW TO AMINOACIDS <GO>
NAME OF ROW 23:PYROGLUTAMIC ACID <GO>
ROW PYROGLUTAMIC ACID COLUMN ENTRY: PYROGLU <GO>
ROW PYROGLUTAMIC ACID COLUMN MAKER: [your name] <GO>
ROW PYROGLUTAMIC ACID COLUMN REFERENCE: ^E <GO>
ROW PYROGLUTAMIC ACID COLUMN RING KEY: ^E <GO>
ROW PYROGLUTAMIC ACID COLUMN STATUS: ^E <GO>
ROW PYROGLUTAMIC ACID COLUMN ABB 1: <GLU <GO>
ROW PYROGLUTAMIC ACID COLUMN ABB 2: EXIT <GO>
23 ROWS
```

MAST

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo
Howard Bilofsky and Harold Perry
Bolt Beranek and Newman Inc.

Summary

MAST supplies a list of all bonded distances, angles, and torsion angles in a molecule.

Calling Sequence

```
RAC PUBLIC MAST
CALL MAST(mol-name)
          <molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

A computed model of the molecule must exist before calling MAST.

MAST works on all types of terminals.

The public procedure MAST types out a list of all bond distances, angles, and torsion angles for a PROPHET molecule (for which a model has been computed). Geometric parameters of hydrogens may optionally be included.

To use MAST, type the command

```
CALL MAST(mol-name)
```

where "mol-name" is the molecule for which you want the parameters listed.

MBLD

by Carl Johnson
Mt. Sinai School of Medicine
and Harold Perry
Bolt Beranek and Newman Inc.

Summary

MBLD provides an alternative method for making molecular models to that available in the PROPHET command language.

Calling Sequence

```
RAC PUBLIC MBLD
CALL MBLD(mol-table)
          <table>
```

Citation

Johnson, C. and Perry, H.; "MBLD" in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

MBLD works on all terminals.

The public procedure MBLD offers an alternative molecular model builder to the PROPHET "COMPUTE MODEL" command.

MBLD constructs a three-dimensional model of a molecule from user provided bond lengths and bond angles.

To use MBLD, type the command

```
     CALL MBLD(mol-table)
                <table>
```

where mol-table is a table containing atom types, bond lengths, bond angles and torsion angles. (The format of this table and suggestions for setting it up can be obtained by asking your PROPHET representative for "MBLD Documentation".)

MBLD uses the information in the table to compute 3D coordinates of all atoms in the molecule; these coordinates may also be optionally stored in the data table mol-table. MBLD then calls %CONNECTMOL (described in part 3 of Chapter III), to fill in the connectivity and set bond types in the new molecule. Finally, MBLD allows you to make a top-level copy of the newly constructed molecule.

NOTE: The principal feature of MBLD, as opposed to the PROPHET Model builder, is that all bond angles and dihedral angles (including those in rings) can be set by the user.

Reference

MBLD is a subset of a FORTRAN program written by Gordon and Pople and available under the title MBLD from the Quantum Chemistry Program Exchange (QCPE).

The following figures illustrate the use of MBLD.

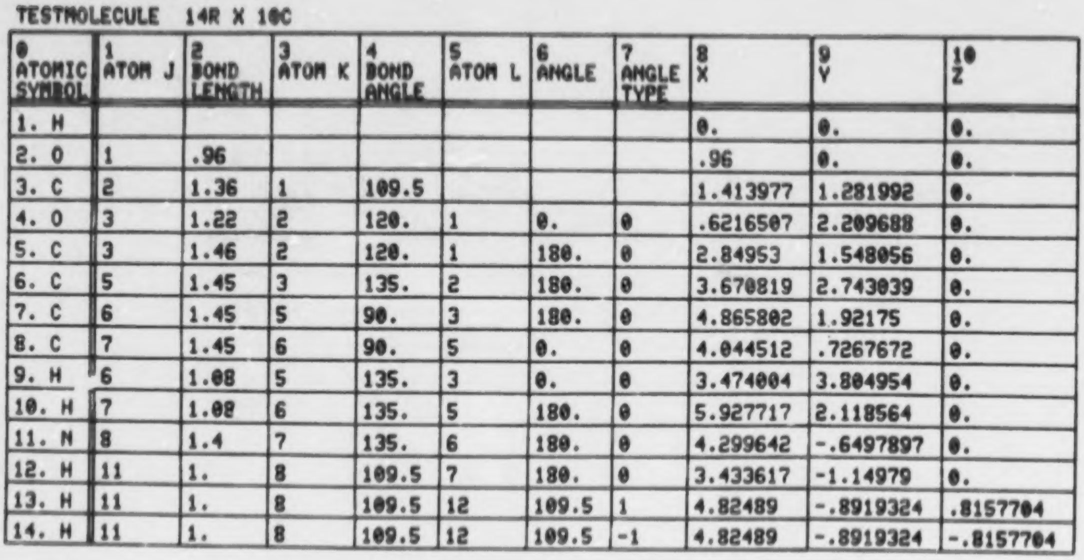
TESTMOLECULE 14R X 7C

0 ATOMIC SYMBOL	1 ATOM J	2 BOND LENGTH	3 ATOM K	4 BOND ANGLE	5 ATOM L	6 ANGLE	7 ANGLE TYPE
1. H							
2. O	1	.96					
3. C	2	1.36	1	109.5			
4. O	3	1.22	2	120.	1	0.	0
5. C	3	1.46	2	120.	1	180.	0
6. C	5	1.45	3	135.	2	180.	0
7. C	6	1.45	5	90.	3	180.	0
8. C	7	1.45	6	90.	5	0.	0
9. H	6	1.08	5	135.	3	0.	0
10. H	7	1.08	6	135.	5	180.	0
11. N	8	1.4	7	135.	6	180.	0
12. H	11	1.	8	109.5	7	180.	0
13. H	11	1.	8	109.5	12	109.5	1
14. H	11	1.	8	109.5	12	109.5	-1

(table before MBLD)

```
>RAC PUBLIC MBLD<
>CALL MBLD(TESTMOLECULE)<

DO YOU WISH TO STORE THE COORDINATES IN YOUR TABLE?YES<


                 <INTERACTION WITH CONNECTMOL>


DO YOU WANT A TOP-LEVEL COPY OF YOUR SIMULATED MOLECULE?YES<
NAME OF MOLECULE AT TOP-LEVEL?MOLQ<

MBLD COMPLETE. TOOK 28.405 CPU SEC.
```

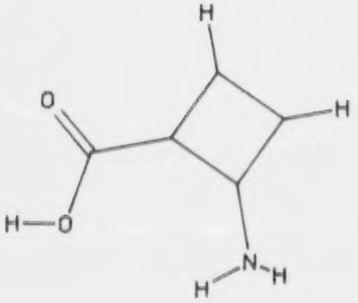
MBLD-4



TESTMOLECULE 14R X 10C

0 ATOMIC SYMBOL	1 ATOM J	2 BOND LENGTH	3 ATOM K	4 BOND ANGLE	5 ATOM L	6 ANGLE	7 ANGLE TYPE	8 X	9 Y	10 Z
1. H								0.	0.	0.
2. O	1	.96						.96	0.	0.
3. C	2	1.36	1	109.5				1.413977	1.281992	0.
4. O	3	1.22	2	120.	1	0.	0	.6216507	2.209688	0.
5. C	3	1.46	2	120.	1	180.	0	2.84953	1.548056	0.
6. C	5	1.45	3	135.	2	180.	0	3.670819	2.743039	0.
7. C	6	1.45	5	90.	3	180.	0	4.865802	1.92175	0.
8. C	7	1.45	6	90.	5	0.	0	4.044512	.7267672	0.
9. H	6	1.08	5	135.	3	0.	0	3.474004	3.804954	0.
10. H	7	1.08	6	135.	5	180.	0	5.927717	2.118564	0.
11. N	8	1.4	7	135.	6	180.	0	4.299642	-.6497897	0.
12. H	11	1.	8	109.5	7	180.	0	3.433617	-1.14979	0.
13. H	11	1.	8	109.5	12	109.5	1	4.82489	-.8919324	.8157704
14. H	11	1.	8	109.5	12	109.5	-1	4.82489	-.8919324	-.8157704

(table after MBLD)



MOLQ displayed THREED COMPLETE

MFBADDBONDS

by Harold Perry
Bolt Beranek and Newman Inc.

Summary

MFBADDBONDS allows the user to interactively add bonds to a molecule. For each bond, the user will be quizzed for the valid names or indices of 2 atoms to be bonded plus the type of the new bond.

Calling Sequence

```
RAC PUBLIC MFBADDBONDS
CALL MFBADDBONDS(MOLQ)
                 <molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

MFBADDBONDS works on all types of terminals.

MFBADDBONDS loops to service user requests for adding bonds. It also warns the user when he has specified an invalid atom.

For each bond, you desire to add to MOLQ, you must supply the names or indices of the atoms to be bonded, plus the bond type. The argument molecule is updated to include the newly specified bond.

To terminate the interaction, type "NO" when prompted with the "NAME 1" request.

The following is an example of the interaction with MFBADDBONDS:

```
 MFBADDBONDS(MOLQ)<
NAME 1?C5<
C5 - NAME 2?C6<
GIVE BOND TYPE (S, T, D, R) C5 - C6?T<
NAME 1?18<
18 - NAME 2?19<
GIVE BOND TYPE (S, T, D, R) 18 - 19?R<
NAME 1?C7<
C7 - NAME 2?C88< C88 NOT FOUND
C7 - NAME 2?C8<
GIVE BOND TYPE (S, T, D, R) C7 - C8?R<
NAME 1?NO<
```

MODIFY

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo
and Howard Bilofsky and Harold Perry
Bolt Beranek and Newman Inc.

Summary

MODIFY permits addition or deletion of atoms and bonds or changes in bond types in order to modify existing molecules or derive new molecules from existing ones.

Calling Sequence

```
RAC PUBLIC MODIFY
CALL MODIFY(old-mol)
            <molecule>
```

Citation

Potter, S., and Weeks, C.; "MODIFY," in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

If atoms of "old-mol" are not named (see the public procedure SETATOMNAMES) then atoms must be referred to by their indices.

MODIFY works on all terminals.

The public procedure MODIFY is designed to make changes to an existing PROPHET molecule or to create a new molecule from an old one.

Possible modifications include: addition or deletion of atoms, addition or deletion of bonds, or changes of bond type.

To use the procedure, the command is

```
CALL MODIFY(old-mol)
```

where "old-mol" is the molecule you want to modify.

You will be asked if you want to make a new, modified molecule, or if you want to make changes on "old-mol" itself.

An interactive quiz is then begun. By typing one of these commands:

```
"ADD ATOM"
"DEL ATOM"
"ADD BOND"
"DEL BOND"
"CHANGE BOND"
```

you will be asked to identify the atoms or bonds to be modified. When finished making changes, type "NONE" when asked for the type of modification to be made.

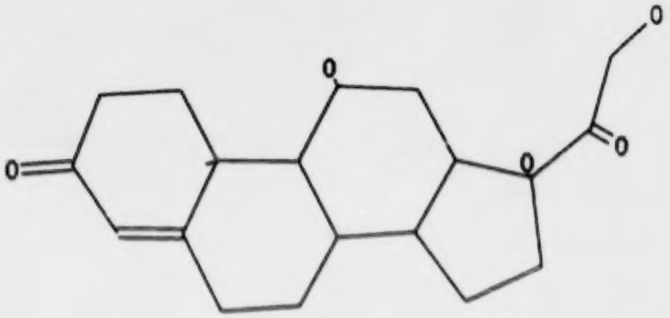
MODIFY will then list a connections summary of the modified molecule, i.e., for each atom in the molecule, its bonds, bondtypes and valence are listed.

Next, MODIFY checks to see that the modified molecule isn't fragmented. If so, it allows you to remodify your molecule.

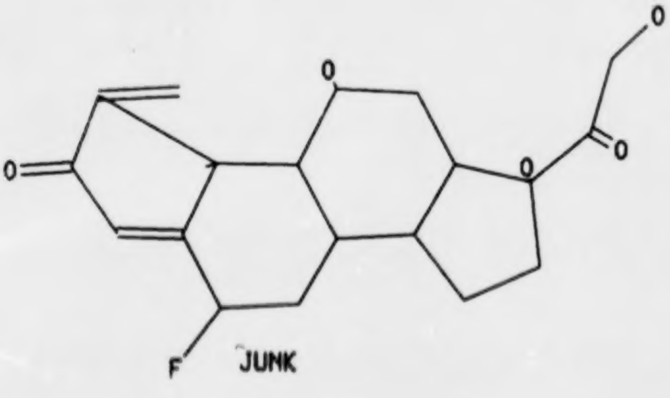
See the following interaction with MODIFY for further explanation.

```
>CALL MODIFY(PR15)<
ARE YOU MAKING A NEW MOLECULE?YES<
 WHAT NAMES SHOULD THE NEW MOLECULE HAVE?  TYPE 'NO' TO TERMINATE
 NAME 1?JUNK< NAME 2?NO<


TYPE 'ADD ATOM', 'DEL ATOM', 'ADD BOND', 'DEL BOND', OR 'CHANGE BOND'
TO INDICATE THE TYPE OF MODIFICATION TO BE MADE.
    TYPE 'NONE' IF NO CHANGE.?DEL ATOM<

 TYPE NAMES OF ATOMS TO BE DELETED.  TYPE 'NO' TO TERMINATE.
 ATOM NAME?H2A< ATOM NAME?H2B< ATOM NAME?H6A< ATOM NAME?NO<


TYPE 'ADD ATOM', 'DEL ATOM', 'ADD BOND', 'DEL BOND', OR 'CHANGE BOND'
TO INDICATE THE TYPE OF MODIFICATION TO BE MADE.
    TYPE 'NONE' IF NO CHANGE.?ADD ATOM<

ARE ATOMS TO BE ADDED BY SPECIFYING THEIR CRYSTALLOGRAPHIC COORDINATES?N<
 TYPE NAMES OF ATOMS TO BE ADDED.  TYPE 'NO' TO TERMINATE.
 ATOM NAME?F6A< ATOMIC SYMBOL?F< DISTANCE F6A-?C6< =?1.45<
 ANGLE F6A-C6-?C5< =?109.< TORSION ANGLE F6A-C6-C5-?C4< =?0.<
 ATOM NAME?NO< GIVE BOND TYPE (S,D,T,R). F6A-C6?S<


TYPE 'ADD ATOM', 'DEL ATOM', 'ADD BOND', 'DEL BOND', OR 'CHANGE BOND'
TO INDICATE THE TYPE OF MODIFICATION TO BE MADE.
    TYPE 'NONE' IF NO CHANGE.?DEL BOND<

TYPE NAMES OF ATOMS FORMING BOND TO BE DELETED.
NAME 1='NO' TO TERMINATE.
 NAME 1?C1< C1-NAME 2?C10< NAME 1?NO<

TYPE 'ADD ATOM', 'DEL ATOM', 'ADD BOND', 'DEL BOND', OR 'CHANGE BOND'
TO INDICATE THE TYPE OF MODIFICATION TO BE MADE.
    TYPE 'NONE' IF NO CHANGE.?ADD BOND<

TYPE NAMES OF ADDITIONAL BONDED ATOMS.  NAME 1='NO' TO TERMINATE.
 NAME 1?C2< C2-NAME 2?C10<
 GIVE BOND TYPE (S,D,T,R). C2-C10?S<


TYPE 'ADD ATOM', 'DEL ATOM', 'ADD BOND', 'DEL BOND', OR 'CHANGE BOND'
TO INDICATE THE TYPE OF MODIFICATION TO BE MADE.
    TYPE 'NONE' IF NO CHANGE.?CHANGE BOND<

TYPE NAMES OF ATOMS AND NEW BOND TYPE (S=SINGLE,D=DOUBLE,T=TRIPLE,
R=AROMATIC) FOR EACH BOND TO BE CHANGED.  NAME 1='NO' TO TERMINATE.
 NAME 1?C1< C1-NAME 2?C2< BOND TYPE (S,D,T,R)?D< NAME 1?NO<


TYPE 'ADD ATOM', 'DEL ATOM', 'ADD BOND', 'DEL BOND', OR 'CHANGE BOND'
TO INDICATE THE TYPE OF MODIFICATION TO BE MADE.
    TYPE 'NONE' IF NO CHANGE.?NONE<
```

```
                              *
INDEX   ATOM        VALENCE     CONNECTIONS     BOND TYPES

1       C1            4.        2 27 28         D S S
2       C2            4.        1 3 10          D S S
3       C3            4.        2 4 22          S S D
4       C4            4.        3 5 29          S D S
5       C5            4.        4 6 10          D S S
6       C6            4.        5 7 30 54       S S S S
7       C7            4.        6 8 31 32       S S S S
8       C8            4.        7 9 14 33       S S S S
9       C9            4.        8 10 11 34      S S S S
10      C10           4.        5 9 19 2        S S S S

              ***** ETC. *****

DO YOU WISH TO MAKE ANY FURTHER MODIFICATIONS?NO< MODIFY COMPLETE
 >
```



PR15



JUNK

*If the valence of an atom is nonstandard, this column entry will have an asterisk next to its value.

MOLECYCLE

by Michael Cory
Burroughs-Wellcome Co.

and Gail Rubin
Bolt Beranek and Newman Inc.

Summary

MOLECYCLE allows you to cycle through the MAKE MOLECULE quiz making each molecule from the one before. The molecules are saved in successive rows of a table you specify.

Calling Sequence

```
RAC PUBLIC MOLECYCLE
CALL MOLECYCLE(table-name, col-num)
                <table>    <fixed>
```

Citation

No citation necessary.

Questions or Comments

Contact Gail Rubin on the PROPHET line.

Things to Remember

MOLECYCLE works only on display terminals with a tablet. MOLECYCLE is not set up to know about SAVED tables; if you use a NAMED SAVED table, you need to fill in the name, maker, and date columns yourself.

MOLECYCLE aids you in filling in the rows of a table with molecules such that each is made from the one in the previous row.

When you call MOLECYCLE, it looks through the column of the table specified by "table-name" and "col-num" to find the molecule before the first EMPTY row.

Then for each of the remaining rows in the column, it invokes the MAKE MOLECULE quiz with the previous molecule. After returning from the quiz, the system puts the new molecule in the next row of the table. Note that MOLECYCLE does not go past the last row of your table. For example, to add three molecules to your table, there must be three empty rows in the table. One way to add these rows is to set any cell in that column to a value (EMPTY can be used).

After each molecule, you are asked if you wish to continue making molecules. MOLECYCLE exits when you answer "NO" or when you have filled in the last row of the table.

MOLEDAT

By Carl Johnson
Mt. Sinai School of Medicine
and Harold Perry
Bolt Beranek and Newman Inc.

Summary

MOLEDAT extracts dihedral angles, bond angles, and bond lengths from a PROPHET molecule or from a set of coordinates. If desired, it also constructs a molecule from a set of coordinates.

Calling Sequence

```
RAC PUBLIC MOLEDAT
CALL MOLEDAT
```

Citation

Johnson, C.; "MOLEDAT", in *PROPHET Molecules* (ed W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

MOLEDAT works <u>only</u> on a display terminal.

The public procedure MOLEDAT allows you to extract dihedral angles, bond angles, and bond lengths from a PROPHET molecule or from a set of coordinates (either Cartesian or X-ray fractional coordinates). (Note: a model must have been computed for a

PROPHET molecule before MOLEDAT can be called.)

After typing the command

CALL MOLEDAT

you must first indicate if the structural data is in the form of a PROPHET molecule or if the coordinates are to be supplied directly; if the later option is chosen, you will be asked whether the data is in cartesian or fractional units and whether it is to be read from a table or the keyboard.

If the coordinates are stored in a table, then the I^{th} row must have the following form:

Column 0 contains the <*text*> TYPE of the I^{th} atom
Column 1 contains the <*float*> X COORDINATE of the I^{th} atom
Column 2 contains the <*float*> Y COORDINATE of the I^{th} atom
Column 3 contains the <*float*> Z COORDINATE of the I^{th} atom

The molecule utility procedure %CONNECTMOL will than be called which allows you to finish the construction of the molecule derived from your coordinate data.

(See the documentation of %CONNECTMOL in part 3 of Chapter III for further explanation if your data is not from a molecule.)

MOLEDAT then displays your molecule with atom indices and repeatedly services requests from the following menu:

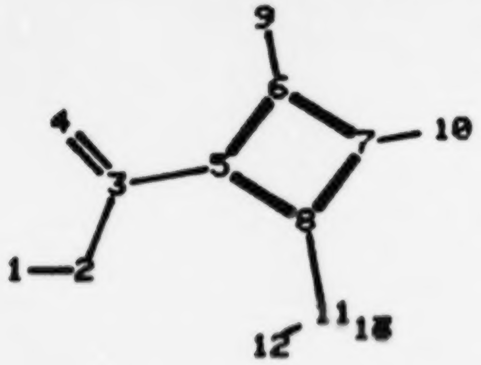
0 = EXIT
1 = CALCULATE DIHEDRAL ANGLE
2 = CALCULATE BOND ANGLE
3 = CALCULATE BOND DISTANCE
4 = LIST INTERNUCLEAR DISTANCES LESS THAN A SPECIFIED CRITICAL VALUE
5 = STORE CARTESIAN COORDINATES IN A TABLE
? = REFRESH SCREEN WITH MOLECULE AND COMMAND LIST

Atoms are to be referenced by specifying their displayed indices.

After exiting this quiz, MOLEDAT gives you the option of saving the constructed molecule if your original data came from the keyboard or a table.

See the following example for further explanation.

```
>CALL MOLEDAT<
SOURCE ((M)ODEL OR (C)OORDINATES) ?   M<
MOLECULE NAME ?  TESTMOL<

COMMANDS ARE:

        0 = EXIT
        1 = CALCULATE DIHEDRAL ANGLE
        2 = CALCULATE BOND ANGLE
        3 = CALCULATE BOND DISTANCE
        4 = LIST INTERNUCLEAR DISTANCES LESS THAN A
            SPECIFIED CRITICAL VALUE
        5 = STORE CARTESIAN COORDINATES IN TABLE
        ? = REFRESH SCREEN WITH MOLECULE & COMMAND LIST
```



```
COMMAND ? 1<     INPUT 4 ATOMS FOR DIHEDRAL ANGLE CALCULATION
    [1]=  5<     [2]=  6<    [3]=  7<    [4]=  8<   TAU = 0.

COMMAND ? 2<     INPUT 3 ATOMS FOR BOND ANGLE CALCULATION
    [1]=  3<     [2]=  5<    [3]=  6<   ALPHA = 135.

COMMAND ? 3<     INPUT 2 ATOMS FOR BOND DISTANCE CALCULATION
    [1]=  3<     [2]=  4<   BOND LENGTH = 1.22

COMMAND ? 4<     INPUT CRITICAL DISTANCE   1.1<
1 --- 2      .96
6 --- 9     1.08
7 --- 10     1.08
11 --- 12     1.
11 --- 13     1.
11 --- 14     1.

COMMAND ? 0<

MOLEDAT COMPLETE. TOOK 11.11301 CPU SEC.
```

NEWMAN

by Mark A. Lavin
Bolt Beranek and Newman Inc.

Summary

NEWMAN displays a Newman projection of a molecule along a specified atom pair.

Calling Sequence

```
RAC PUBLIC NEWMAN
CALL NEWMAN(mol-name,  atom1,  atom2)
          <molecule> <fixed> <fixed>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

NEWMAN only works on display terminals. The molecule must have a 3D Model.

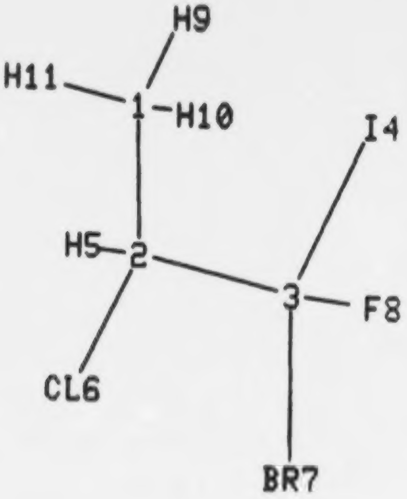
NEWMAN is used to display a Newman projection of a molecule along a line specified by two atoms. Specifically, the command

```
CALL NEWMAN(MOL,A1, A2)
```

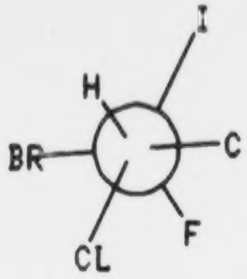
will produce the Newman projection looking from atom A1 toward atom A2 in molecule MOL. The bonds/atoms immediately adjacent to A1 are shown as labeled lines piercing the circle, representing A1/A2, while the bonds/atoms immediately adjacent to A2 are shown as labeled lines running into, but not piercing, the circle (see illustrations below). The relative directions of the atom/bond lines reflect the three-dimensional structure of the molecule in the A1/A2 vicinity.

Note that the atoms A1 and A2 must be specified by their numeric index, so it may be convenient to display the molecule using the ATOMINDEX display option before using NEWMAN.

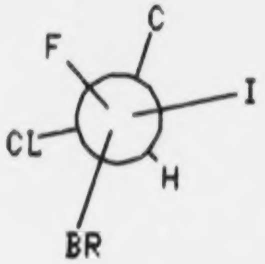
Consult the following example for further illustration.


H9
H11
1
H10
I4
H5
2
3
F8
CL6
BR7

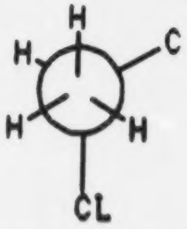
DISPLAY FOO THREED
COMPLETE ATOMINDEX



CALL NEWMAN(FOO,2,3)



CALL NEWMAN(FOO,3,2)



CALL NEWMAN(FOO,1,2)

ORMOL

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo
and Howard Bilofsky and Mark Lavin
Bolt Beranek and Newman Inc.

Summary

ORMOL allows a molecule to be oriented so that a specified atom is at the origin, a second atom is on the X-axis, and two additional atoms are used to define a vector parallel to the XY plane.

Calling Sequence

```
RAC PUBLIC ORMOL
CALL ORMOL(mol-name)
          <molecule>
```

Citation

Potter, S. and Weeks, C.; "ORMOL", in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Gail Rubin on the PROPHET line.

Things to Remember

ORMOL works on all terminals, and requires that your molecules have a model.

The public procedure ORMOL allows you to orient a molecule so that a specified atom is at the origin, a second atom is on the X-axis, and two additional atoms are used to define a vector parallel to the XY plane (the plane of the screen).

Standard orientations for steroids and thyroactive compounds are included as alternatives.

To use ORMOL, type the command

```
CALL ORMOL(mol-name)
```

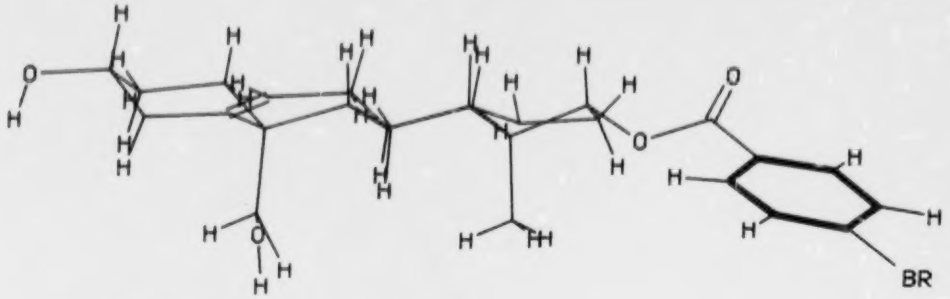
where "mol-name" is the molecule you want to orient.

You will then be asked to designate the type of orientation to be used.

ORMOL allows the following orientations:

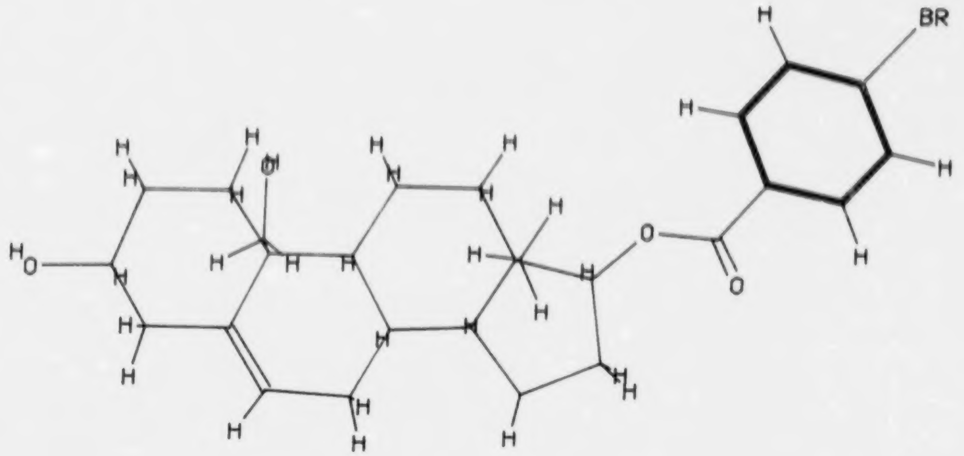
ORIENTATION TYPE	ORIGIN ATOM	ATOM ON X-AXIS	ATOMS PARALLEL TO X-Y PLANE	ADDITIONAL ROTATION
STER	C10	C13	C14, C12	-
THYR	4	1	3, 5	65° about x axis
PG	C9/C11	C11	C8, C12	-
BV	C3	C4	C3, C2	-
USER	<as specified by the user>			

In the following example, the molecule AN55 is reoriented to the steroid orientation, by using ORMOL.



(AN55 displayed THREED COMPLETE before ORMOL)

```
>ORMOL(AN55)<
ORIENTATION       ORIGIN   ATOM ON  ATOMS || TO    ADDITIONAL
TYPE              ATOM     X-AXIS   X-Y PLANE      ROTATION
STER              C10      C13      C14, C12          --
THYR              4        1        3, 5           65 degrees about x-axis
PG                C9/C11   C11      C8, C12           --
BV                C3       C4       C3, C2            --
USER              <as specified by the user>

TYPE NAME OF STANDARD ORIENTATION TO BE USED. ?STER<
ORMOL COMPLETE. TOOK 2.349998 CPU SEC.
```



(AN55 displayed THREED COMPLETE in the STEROID orientation)

PLANE

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo

Summary

PLANE creates a table containing the distances of all atoms in a molecule from a least squares plane passed through a specified set of atoms.

Calling Sequence

```
RAC PUBLIC PLANE
CALL PLANE
```

Citation

Potter, S. and Weeks, C.; "PLANE", in *PROPHET Molecules* (ed. W.P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April 1980.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

PLANE works on all types of terminals.
PLANE works only for molecules with models.

The public procedure PLANE computes the distances of all atoms in a molecule from a least squares plane passed through a specified set of atoms.

The molecules to be entered may be individual molecules, or they may be molecules in an array or a table.

To use PLANE, type the command

```
CALL PLANE.
```

You will be asked for a variable name which contains molecules. This can be a table containing molecules, an array of molecules, or an individual molecule. More than one molecule table can be used.

You will be asked to designate the atoms defining the least squares plane. Your response must be of the form:

"ATOM M to ATOM N"
or "ATOM M, ATOM N"
or a list consisting of these forms,
e.g. "C3 TO C9, C11, C12"

Atom names or atom indices may be specified in your response.

You will then be asked for the name of the table to contain the computed values. In the output table, components of the eigenvectors defining the least squares plane are placed in the first 3 rows of the tables; the root-mean-square deviation (RMS) of the atoms included in the least squares plane is in row 4.

Atoms included in the least-squares plane are flagged with asterisks (*).

See the example below for an illustration of using PLANE.

MOLECULES 6R X 1C

0 ROWNAME	1 FORMULA
1. AN65	$C_{20}H_{32}O_3$
2. AN78	$C_{20}H_{32}O_3$
3. PR81	$C_{22}H_{31}NO_2$
4. PR103	$C_{23}H_{38}O_2$
5. PR104	$C_{22}H_{35}O_2$
6. PR113	$C_{23}H_{38}N_2$

(table supplied to PLANE)

```
>PLANE<
TYPE NAME OF VARIABLE CONTAINING MOLECULES?MOLECULES<
IF ADDITIONAL MOLECULE TABLES ARE TO BE SPECIFIED, TYPE TABLE NAME.
  OTHERWISE, TYPE "NONE".  TABLE NAME?NONE<
ARE ALL MOLECULES TO BE INCLUDED IN THE CALCULATIONS?N<

TYPE THE NAMES OF THE MOLECULES TO BE CONSIDERED.
  TYPE "NONE" TO TERMINATE.
    MOLECULE NAME?AN65<
    MOLECULE NAME?PR104<
    MOLECULE NAME?NONE<
TYPE "C" OR "R" TO INDICATE WHETHER MOLECULE REGISTRY NUMBERS ARE
TO BE USED AS COLUMN HEADINGS OR ROW NAMES?C<

TYPE THE NAMES OR INDICES OF THE ATOMS DEFINING THE
   LEAST-SQUARES PLANE?3 TO 11<
 WHAT NAME DO YOU WANT FOR THE OUTPUT TABLE?MDIST<

ARE CALCULATIONS TO BE DONE FOR HYDROGEN ATOMS?N<




PLANE COMPLETE.  TOOK 26.223 CPU SEC.
>
```

```
MDIST  30R X 2C
0              1       2         0             1        2
ROWNAME        AN65    PR104     ROWNAME       AN65     PR104
 1. B11        -.049   -.068     20. C16       -.430    -.41
 2. B12        -.053   -.058     21. C17       -.704    -.694
 3. B13        .997    .996      22. C18       1.505    1.539
 4. RMS        .287    .268      23. C19       1.627    1.806
 5. C1         -.852   -.5       24. C20       2.782    -.493
 6. C2         -.596   -.171     25. O3B       -.041    -.069
 7. C3 *       -.498   -.441     26. O17B      -.304
 8. C4 *       .45     .377      27. O19       1.603
 9. C5 *       .23     .169      28. C21                -1.447
10. C6 *       -.072   -.097     29. C22                -.78
11. C7 *       -.283   -.239     30. O20                .838
12. C8 *       .191    .191
13. C9 *       -.256   -.273
14. C10 *      .222    .321
15. C11 *      .017    -.008
16. C12        -.545   -.527
17. C13        .002    .015
18. C14        -.411   -.379
19. C15        -.138   -.102
```

(output table produced by PLANE)

PRECONNECT

by Steven F. Watkins
Louisiana State University

Summary

PRECONNECT takes as input X, Y, Z atom coordinates (cartesian or crystallographic) and turns them into a PROPHET molecule by using MOLCONNECT.

Calling Sequence

```
RAC PUBLIC PRECONNECT
CALL PRECONNECT
```

Entry

```
CALL PRECONNECTQT(coords-tab, mol-name)
                  <table>      <text>
```

Questions or Comments

Contact Gail Rubin on the PROPHET line.

Things to Remember

PRECONNECT works on all terminals. It calls the public procedure MOLCONNECT, which calls the system procedure %CONNECTMOL. You should read the documentation for MOLCONNECT in this chapter and for %CONNECTMOL in Chapter III, Part 3.

PRECONNECT is an interactive program which accepts XYZ coordinates as you type them in. PRECONNECTQT accepts a table of atom types, names, and XYZ coordinates that has already been prepared.

When you call PRECONNECT, you are asked several questions:

1. WHAT IS THE NAME OF YOUR MOLECULE? The name you give here will be the name of your <u>new</u> molecule. The table that PRECONNECT makes with the coordinates you specify will be a PRIVATE table named mol-name.'COORDS' (e.g., if your molecule is to be named MOL1, the table name will be MOL1COORDS).

2. ABOUT HOW MANY ATOMS? Estimate how many atoms there are in your molecule. This number is only used for efficiently allocating space; it does not have to be exact.

3. INPUT A SCALE FACTOR FOR THE INPUT XYZ. FOR EXAMPLE, A VALUE OF 0.0001 WOULD ALLOW YOU TO ENTER "-0.0347" as "-347" AND AVOID "." SCALE FACTOR? Enter a number. For no scaling, type 1.

Next comes the atom input quiz. You must specify all atoms of one type before you specify atoms of another type (e.g., specify all carbons, then all hydrogens, etc.).

4. ENTER ATOM TYPE (C, H, FE, ETC.) OR <CTRL-E>? Specify the atomic symbol for the element you wish to have atoms of. (When you are done with all atoms, type <CTRL-E>). As an example, suppose you responded BR. You would then be asked the following questions.

5. X OR <CTRL-E> FOR BR1? Input the X coordinate.

6. Y? Type the Y coordinate.

7. Z? Give the Z coordinate.

5. X OR <CTRL-E> for BR2? and so on. PRECONNECT keeps expecting XYZ coordinates for atoms of this type until you type <CTRL-E> to go on to other types. If you answer <CTRL-E>, the next question is:

4. ENTER ATOM TYPE... After the last atom has been entered, answer <CTRL-E> to questions 5 and 4.

8. DO YOU WANT TO EDIT THE TABLE? If you answer NO, PRECONNECT proceeds to question 9. If you answer YES, it types a message, displays your table and <u>exits from PRECONNECT</u>. The message tells you how to resume execution (with PRECONNECTQT). You can change values in the table with normal PROPHET commands (EDIT, SET). To resume making the molecule, you would type something like:

```
CALL PRECONNECTQT(MOL1COORDS,'MOL1')
```

9. DOES YOUR TABLE CONTAIN CARTESIAN COORDINATES? If you answer YES, the procedure proceeds to question 11. If you answer NO, the program displays the previous lattice constants used (if any).

10. DO YOU WISH TO USE THE PREVIOUS LATTICE CONSTANTS? If there are no previous constants, or you wish to change them, answer NO and you will be asked for the new constants (A, B, C, ALPHA, BETA, and GAMMA). These are the standard crystal lattice parameters, and as you enter them they are stored in your table in footnotes for later use.

At this point, a temporary table is made from your coordinates table so that transformation to cartesian coordinates or translation resulting from moving the origin will not affect your private (permanent) table.

11. DO YOU WANT ORIGIN AT CENTER OF MOLECULE? If you answer YES, the coordinates in the temporary table are adjusted so that (0,0,0) is at the center of the molecule.

12. DO YOU WANT TO SEE THE TABLE BEFORE THE MOLECULE IS CREATED? The temporary table is displayed if you answer YES.

Once PRECONNECT has done all these preparations, the last thing it does is call MOLCONNECT with information from the temporary table. (See the CONNECT documentation for further information.)

One other note on using PRECONNECTQT: If you have a table containing one row for each atom already made up with your atomic coordinates, it must be in the form which PRECONNECT expects. This form is:

column 1	TEXT	atom type
column 2	TEXT	atom name
column 3	NUMBER	x coordinates
column 4	NUMBER	y coordinates
column 5	NUMBER	z coordinates

Additional columns are ignored.

REFINE

by Douglas
Medical Foundati

Summa

REFINEMOL carries out
three-dimensional model of a
minimization program.

Calling S

RAC PUBLIC REFINEMOL
CALL REFINEMOL

Citat

No citation necessary.

Questions or

Send message to 'ROHRER' in

Things to

The molecule whose model is
your PRIVATE SAVED table.

REFINEMOL works on all types

The public procedure REFIN
minimization program to refine
molecule stored in your PRIVATE SAV
system procedure %AUTOBAT to schedu
time that you specify. The energy
is described in more detail in
procedure EDITMODEL in this chapter

To use REFINEMOL, type the com

```
CALL REFINEMOL
```

REFINEMOL will proceed to ask a ser

1. WHAT IS THE MOLECULE'S NAME?
   the molecule is saved in you

OL

Rohrer
n of Buffalo

Y

the refinement of the
olecule via the QCFF energy

quence

on

Comments

ROPHET.

emember

o be refined must be stored in

of terminals.

MOL uses the QCFF energy
he three-dimensional model of a
D table. REFINEMOL uses the
e QCFF to run in batch mode at a
inimization program that is used
the description of the public

and

s of questions.

ply with the name under which
PRIVATE SAVED table. After you

answer this question, REFINEMOL types a reminder that your molecule must be saved in your PRIVATE SAVED table.

2. SHOULD HYDROGEN ATOMS BE INCLUDED IN THE REFINEMENT? Answer YES or NO.

3. SHOULD THE DATA FILES BE DELETED AT THE END OF THIS REFINEMENT? Answer NO if you wish to examine the data files that QCFF writes in the course of carrying out the refinement. At this point REFINEMOL calls the system procedure %AUTOBAT, which asks questions 4 to 6.

4. ABOUT HOW MANY CPU MINUTES DO YOU THINK THE JOB WILL TAKE TO COMPLETE (IT'S BETTER TO SLIGHTLY OVERESTIMATE) ? Answer with the number of minutes you are willing to let QCFF work on carrying out the refinement.

5. WHAT DAY DO YOU WANT THE JOB RUN (TODAY = 0; TOMORROW = 1; ETC.)? Answer with the appropriate integer.

6. WHAT HOUR (EASTERN TIME;) DO YOU WANT THE JOB TO START (TYPE 9:15 AM, 11:37 PM, ETC.)? Answer with an appropriate clock hour. You should never schedule QCFF to run between 9 am and 6 pm Eastern Time, since it is computationally intensive. In fact, it is cheapest and most appropriate to schedule it for late night hours.

At the time you specify, QCFF will run to refine your molecule, and then save the refined molecule back into your PRIVATE SAVED table.

SETATOMNAMES

by Stephen Potter and Charles Weeks
Medical Foundation of Buffalo
and Howard Bilofsky and Harold Perry
Bolt Beranek and Newman Inc.

Summary

SETATOMNAMES allows you to give atoms of a molecule user defined names.

Calling Sequence

```
RAC PUBLIC SETATOMNAMES
CALL SETATOMNAMES(mol-name)
                  <molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

SETATOMNAMES works on all types of terminals.

Some of the MFB public molecule procedures assume that atoms of a molecule have been given a name (e.g., C1, C2, H1A).

The public procedure SETATOMNAMES allows you to name atoms in one of your molecules.

To use SETATOMNAMES, type the command

```
CALL SETATOMNAMES(mol-name)
```

where "mol-name" is the name of the molecule whose atoms you want to name.

You will be asked to supply the index of the atom, and the name that atom is to have. When you are through supplying names, answer "0" when asked for the atom index.

It is probably advisable to display your molecule INDEXed before using this procedure.

SETMOLNAMES

by Harold Perry

Bolt Beranek and Newman Inc.

Summary

SETMOLNAMES allows you to give new names to, or change the names of a PROPHET molecule.

Calling Sequence

```
RAC PUBLIC SETMOLNAMES
CALL SETMOLNAMES(mol-name)
                 <molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Harold Perry on the PROPHET line.

Things to Remember

SETMOLNAMES works on all types of terminals.

Just after you type "EXIT" in the MAKE MOLECULE quiz, you are given a chance to supply names for a molecule.

The public procedure SETMOLNAMES offers you an easy way to give a molecule these names at some later time. It can also be used to delete or change names you have already supplied.

Note that some of the Medical Foundation of Buffalo molecule procedures assume that a molecule has at least two names: the first a structure identifier; the second, the compound's IUPAC name.

To use SETMOLNAMES, the command is

```
CALL SETMOLNAMES(mol-name)
```

where "mol-name" is the molecule you want to name.

Current names, if any, will be typed out and you can then ADD names (type "A"), REPLACE names (type "R"), or LEAVE things as they are (type "L"). If you want to REPLACE a name, you are asked for the number of the name to be replaced. Supply the name when prompted.

SETPEPTIDEBONDS

by Wayne Rindone
Bolt Beranek and Newman Inc.

Summary

SETPEPTIDEBONDS searches a molecule for amide bonds and sets appropriate bond length, bond angle and dihedral angle constraints for the atoms and bonds involved in any amide bonds that are found. For the special case where one or more of the amide bonds found are, in fact, peptide bonds in a polypeptide chain, SETPEPTIDEBONDS also sets dihedral angle constraints for the phi and psi torsion angles in the backbone of the molecule.

Calling Sequence

```
RAC PUBLIC SETPEPTIDEBONDS
CALL SETPEPTIDEBONDS(mol-name, phi-angle, psi-angle)
                     <molecule>  <float>    <float>
```

Citation

No citation necessary.

Questions or Comments

Contact Wayne Rindone on the PROPHET line.

Things to Remember

SETPEPTIDEBONDS works on all terminals. SETPEPTIDEBONDS merely sets appropriate constraints for the amide bonds in the molecule; it does not itself calculate the three-dimensional model of the molecule. To do this, you must use the PROPHET COMPUTE MODEL command after running SETPEPTIDEBONDS.

The PROPHET molecular model builder does a credible job of creating a three-dimensional conformation for small organic compounds that contain simple bonding patterns. However, it is not capable of recognizing intermediate bonding patterns such as those involved in an amide bond, illustrated in Figure 1.

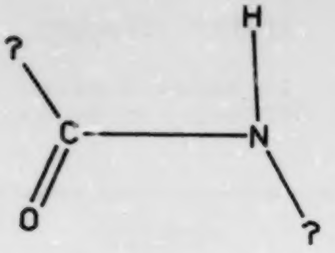


Figure 1. Amide bond.

Since the single bond between the carbon and nitrogen atoms in the amide bond structure has some double bond character, and the double bond between the carbon and oxygen atoms has some single bond character, you must set some appropriate constraints (see Chapter II, Part 2) before typing the COMPUTE MODEL command in order to achieve a reasonable three-dimensional conformation.

It is particularly important to set the relevant constraints before computing the model of a molecule containing the special case of the amide bond known as a peptide bond, illustrated in Figure 2.

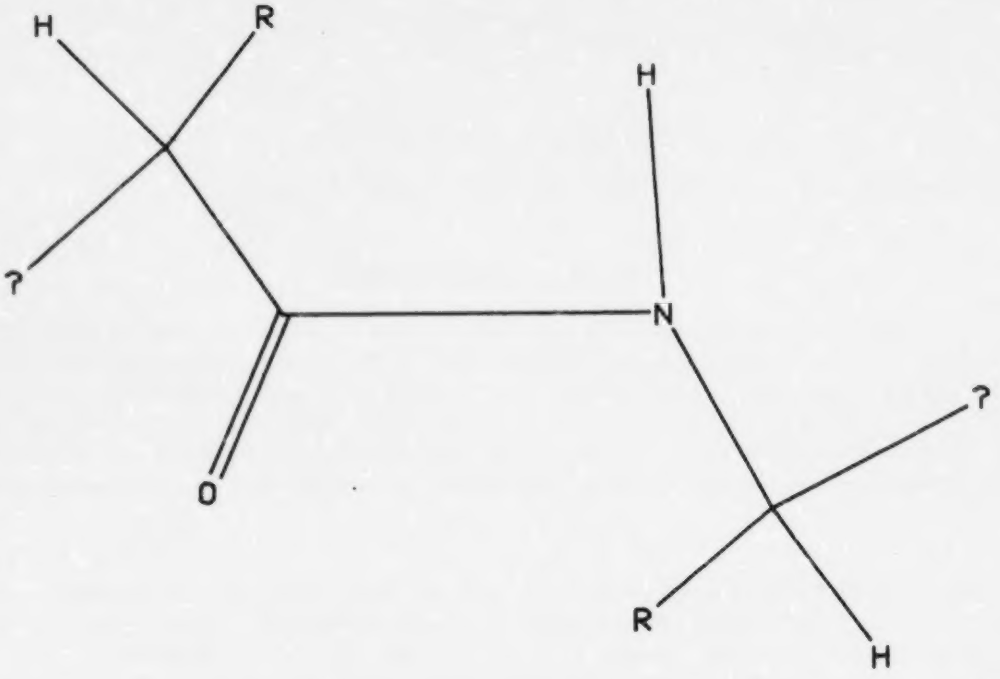


Figure 2. Peptide bond.

Since peptide bonds are often arranged in long linear sequences (as in polypeptides), even small errors in the calculation of bond lengths and angles in a single peptide bond tend to accumulate and propagate over the entire length of the molecule, resulting in major inaccuracies in the molecule's secondary structure.

The public procedure SETPEPTIDEBONDS automates the setting of appropriate constraints for amide bonds and peptide bonds. The values used for the constraint settings are based on those in Dickerson, R. E. and I. Geis (1969) <u>The Structure and Action of Proteins</u>, Harper and Row.

To use SETPEPTIDEBONDS, type the command

```
CALL SETPEPTIDEBONDS(mol-name,phi-angle,psi-angle)
```

where "mol-name" is the molecule you wish to analyze. If any peptide bonds are present, you can provide values for "phi-angle" and "psi-angle", which are the dihedral angles in degrees about the two freely rotating polypeptide backbone bonds adjacent to each peptide bond. SETPEPTIDEBONDS sets the same phi and psi dihedral angle constraints for <u>every</u> peptide bond it finds in the molecule. Some of the more common combinations of phi and psi angles in biological materials are stored in the public PROPHET table PHIPSIANGLES, which is illustrated in Figure 3.

PHIPSIANGLES VALUES OF PHI AND PSI TORSION ANGLES FOR SOME STANDARD
POLYPEPTIDE CONFORMATIONS ACCORDING TO THE INTERNATIONAL UNION
OF BIOCHEMISTRY, BIOCHEMICAL NOMENCLATURE AND RELATED DOCUMENTS
(THE BIOCHEMICAL SOCIETY, 1978) 11R X 4C

0 ROWNAME	1 PHI ANGLE (DEGREES)	2 PSI ANGLE (DEGREES)	3 OMEGA ANGLE (DEGREES)	4 REFERENCE
1. RIGHT-HANDED ALPHA HELIX	-57.	-47.	180.	ARNOTT, S., AND DOVER, S.D.(1967),
2. LEFT-HANDED ALPHA HELIX	57.	47.	180.	ARNOTT, S., AND DOVER, S.D.(1967),
3. 3SUB10 HELIX	-49.	-26.	180.	RAMAKRISHNAN, C., AND RAMACHANDRAN, G.N. (1965)
4. ALTERNATE 3SUB10 HELIX	-74.	-4.	180.	SCHELLMAN, J.A., AND SCHELLMAN, C. (1964)
5. 2SUB7 RIBBON	-75.	70.	180.	RAMAKRISHNAN, C., AND RAMACHANDRAN, G.N. (1965)
6. POLYPROLINE HELIX	-78.	149.	180.	ARNOTT, S., AND DOVER, S.D. (1968)
7. ANTI-PARALLEL PLEATED SHEET	-139.	135.	-178.	ARNOTT, S., DOVER, S.D., AND ELLIOTT, A. (1967)
8. PARALLEL PLEATED SHEET	-119.	113.	180.	SCHELLMAN, J.A., AND SCHELLMAN, C. (1964)
9. POLYGLYCINE II	-80.	150.	180.	RAMACHANDRAN, G.N., SASISEKHARAN, V., AND RAMAKRISHNAN, C.-
10. PI HELIX	-60.	-60.	180.	RAMAKRISHNAN, C., AND RAMACHANDRAN, G.N. (1965)
11. FULLY EXTENDED CHAIN	180.	180.	180.	

Figure 3. PUBLIC PHIPSIANGLES.

As a first simple example, suppose we have a molecule called AMIDE, which is illustrated in Figure 4 with atom indices and bond indices. Typing the statement

```
CALL SETPEPTIDEBONDS(AMIDE, , )
```

results in setting up the CONSTRAINTS table of AMIDE illustrated in Figure 5.

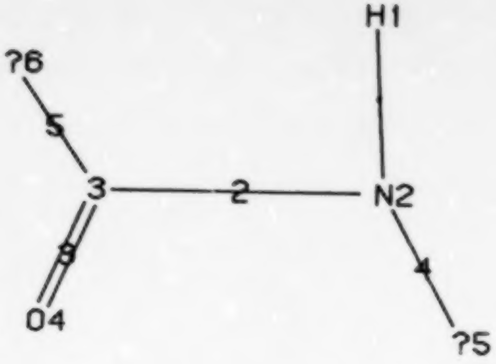


Figure 4. AMIDE displayed ATOMINDEX BONDINDEX.

CONSTRAINTS 7R X 4C

	1 TYPE	2 ATOMS	3 BONDS	4 VALUE
1	BONDLENGTH		[5]	1.51
2	BONDLENGTH		[3]	1.24
3	BONDLENGTH		[2]	1.325
4	BONDLENGTH		[4]	1.455
5	TORSION		[3,2,1]	180.
6	BOND ANGLES	[3]	[3,5,2]	[120.5,116.,123.5]
7	BOND ANGLES	[2]	[4,1,2]	[116.,122.,122.]

Figure 5. CONSTRAINTS OF AMIDE after calling SETPEPTIDEBONDS.

We may look at a similar case for a molecule called PEPTIDE, illustrated in Figure 6. The commands

```
CALL SETPEPTIDEBONDS(PEPTIDE,-139.,135.)
```

results in setting up the CONSTRAINTS table of PEPTIDE shown in Figure 7.

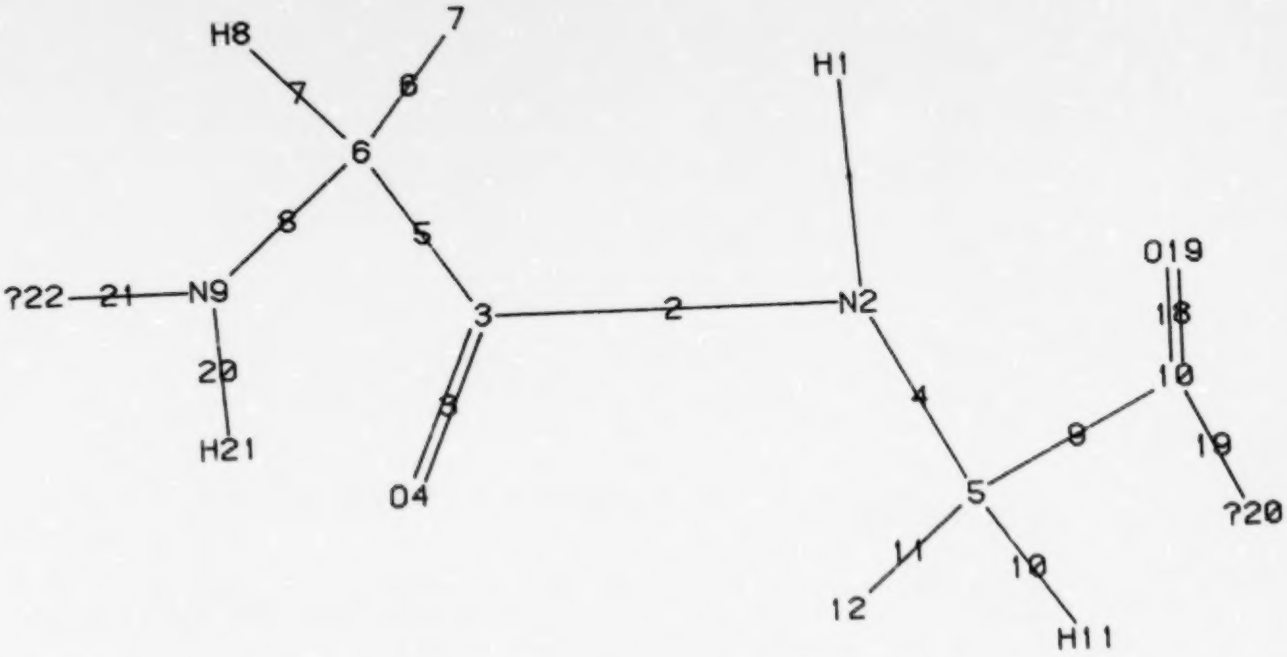
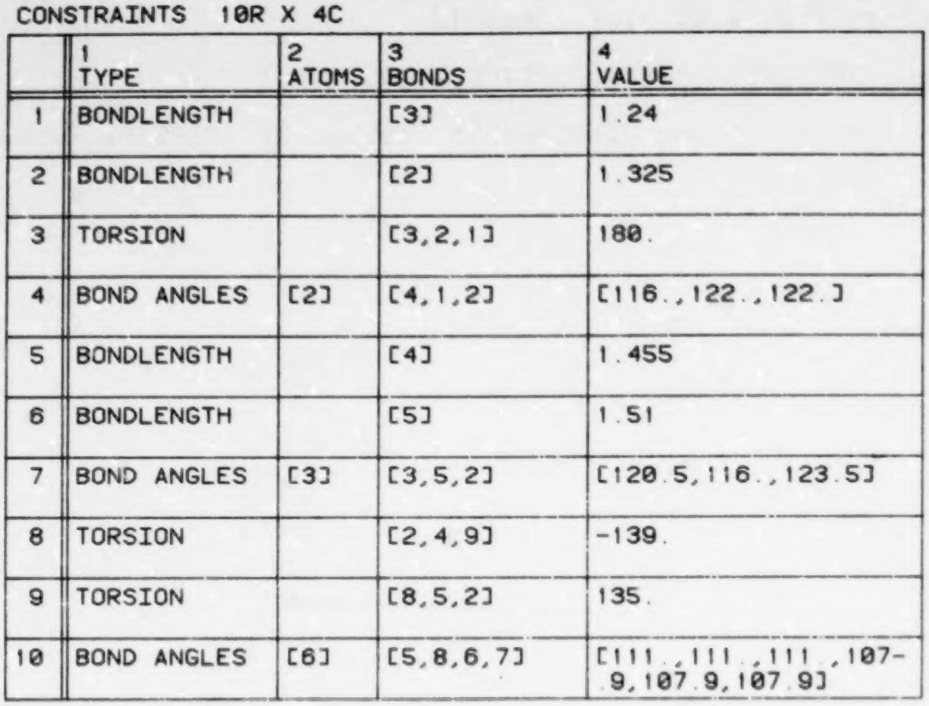

H8
7
7
6
6
H1
8
5
O19
?22
21
N9
3
2
N2
18
20
3
4
10
9
19
H21
O4
5
?20
11
10
12
H11

Figure 6. PEPTIDE displayed ATOMINDEX BONDINDEX.



CONSTRAINTS 10R X 4C

	1 TYPE	2 ATOMS	3 BONDS	4 VALUE
1	BONDLENGTH		[3]	1.24
2	BONDLENGTH		[2]	1.325
3	TORSION		[3,2,1]	180.
4	BOND ANGLES	[2]	[4,1,2]	[116.,122.,122.]
5	BONDLENGTH		[4]	1.455
6	BONDLENGTH		[5]	1.51
7	BOND ANGLES	[3]	[3,5,2]	[120.5,116.,123.5]
8	TORSION		[2,4,9]	-139.
9	TORSION		[8,5,2]	135.
10	BOND ANGLES	[6]	[5,8,6,7]	[111.,111.,111.,107.9,107.9,107.9]

Figure 7. CONSTRAINTS OF PEPTIDE after calling SETPEPTIDEBONDS.

The following is a more comprehensive example of the use of SETPEPTIDEBONDS. Let us suppose we have sketched in (using MAKE MOLECULE) a molecule called PENTAGLYCINE, as shown in Figure 8.

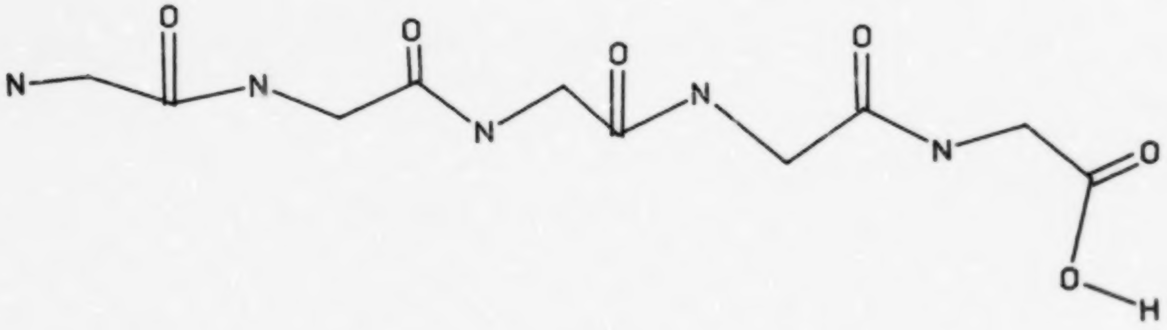


Figure 8. PENTAGLYCINE sketch.

Our goal is to compute a conformation for PENTAGLYCINE that approximates the conformation present in a beta-pleated sheet. We therefore type the commands:

```
RAC PUBLIC SETPEPTIDEBONDS<GO>
CALL SETPEPTIDEBONDS(PENTAGLYCINE,-139.,135.)<GO>
COMPUTE MODEL OF PENTAGLYCINE<GO>
```

The PENTAGLYCINE conformation that results is displayed THREED COMPLETE in Figure 9.

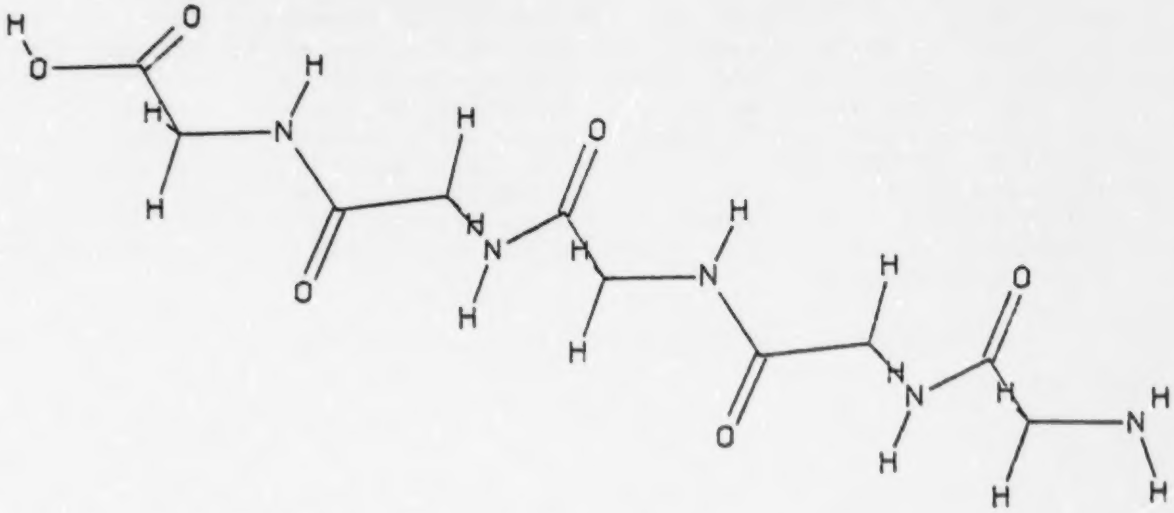


Figure 9. PENTAGLYCINE displayed THREED COMPLETE after calling SETPEPTIDEBONDS and using COMPUTE MODEL.

Notes

1. The COMPUTE MODEL command in PROPHET does not take constraints into account when they pertain to atoms and bonds in ring structures. Although SETPEPTIDEBONDS may set constraints for peptide bonds in cyclic polypeptides, they will be ignored when the model is computed as long as the ring is closed.

2. SETPEPTIDEBONDS sets the phi and psi dihedral angle constraints to the same values for all peptide bonds found in the entire molecule. After calling SETPEPTIDEBONDS, you may use any of the following to modify the constraint values where appropriate: the CONSTRAIN command (see Chapter II, Part 2); the system procedure %SETCONVALUE (see Chapter III, Part 2); or the system procedure %SETCONTORSION (see Chapter III, Part 3).

3. SETPEPTIDEBONDS assumes that all the amino acids linked by the peptide bonds it finds are their usual L-isomers. If there are any D-amino acids present, you may modify the order of the bond array stored for the bond angle constraints about the alpha carbon of the amino acid. For example, in Figure 7 constraint 10 establishes bond angle constraints about the alpha carbon (atom 6) in PEPTIDE. The bonds in that constraint are given in the order [5, 8, 6, 7]. Changing the order to [5, 8, 7, 6] would result in a D-amino acid when you type the COMPUTE MODEL command. Changing the order of the bonds could be accomplished with the statement

```
CALL %SETCONBONDS(PEPTIDE,10,[5,8,7,6])
```

SPACEMOL

by Doug Rohrer
Medical Foundation of Buffalo

and Eric Meyer
Bolt Beranek and Newman Inc.

Summary

SPACEMOL displays a space-filling model of a molecule.

Calling Sequence

```
CALL SPACEMOL(mol-name)
              <molecule>
```

Citation

Rohrer, D.; "SPACEMOL" in PROPHET Molecules (ed. W. P. Rindone and T. Kush); Bolt Beranek and Newman Inc., Cambridge, MA; April, 1980.

Questions or Comments

Contact Eric Meyer on the PROPHET line.

Things to Remember

Because of the complexity of the computational problems in deriving the space-filling model, SPACEMOL uses a large amount of computer time, even for small molecules.

Model coordinates must be assigned to the molecule before calling SPACEMOL.

SPACEMOL works on display terminals only.

SPACEMOL displays a molecule as a set of interlocking spheres, with each sphere representing one atom. The position of each atom is determined by its model coordinates, and its relative size is given by its van der Waal's radius.

Optionally, a line drawing of the molecule (a "stick" figure) may be displayed along with the space-filling model. The stick figure is similar to the regular PROPHET molecule display THREED COMPLETE. All atoms are labeled by type ("C", "H", etc.); however, all bonds are shown as single lines regardless of their type (single, double, triple, or resonant). The stick display is shown in the same position and orientation as the space-filling model.

SPACEMOL is invoked by typing

```
CALL SPACEMOL(mol-name)
```

The procedure first asks you

```
TYPE OF DRAWING:  STICK(ST), SPHERE(SP), OR BOTH(BO)?
```

If you answer "STICK", only the simple stick drawing is shown. This can be useful to ensure that the molecule fits on the screen. If you respond "SPHERE", only the space-filling model is shown with no atoms labeled and no bonds indicated. Responding "BOTH" gives you the two displays together, the stick figure appearing to hover within the space-filling model like a skeleton within a body.

If you respond to the first question with "SPHERE" or "BOTH", the next question is

```
ENTER SPHERE SIZE?
```

This allows you to scale the atomic radii relative to the bond lengths of the molecule. Responding CTL-S (control-S) or 1 causes the radii to be given by the van der Waal's radius of each atom. Responding with a 2 doubles the radii of the atoms (relative to bond length), .5 halves them, and so forth. If the sphere size is close to zero, the atoms will no longer touch one another, and the resulting display will be similar to the ball and stick display of the %PERSPECTIVE procedure.

SPACEMOL also asks you to

```
ENTER SCALE FACTOR, CM./ANGSTROM?
```

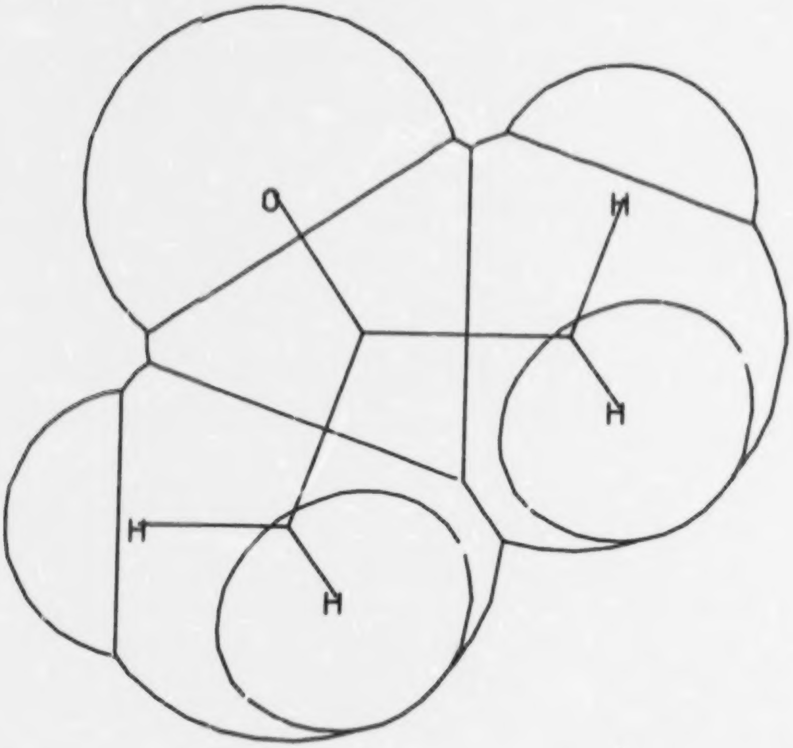
If you simply hit <GO>, a default scale factor of 1 is assumed. Otherwise, the display is scaled according to the number you give it.

After asking these questions, SPACEMOL proceeds to display the molecule. The display is centered on the screen and oriented according to the current orientation of the molecule's model. Thus, if you want to see the molecule again from a different orientation, you must 1) **DISPLAY** it **THREED**, 2) **TURN** it to the desired orientation, 3) **FREEZE** it to keep that orientation, and 4) call SPACEMOL again.

EXAMPLE

Suppose you wish to use SPACEMOL on the ACETONE molecule. ACETONE must already have had its model computed. You can proceed as follows:

```
>CALL SPACEMOL(ACETONE)<
TYPE OF DRAWING: STICK(ST), SPHERE(SP), OR BOTH(BO)?BO<
ENTER SPHERE SIZE ?S^<
ENTER SCALE FACTOR, CM./ANGSTROM ?2<
```



STEREOBOX

by Michael Cory and Robe
Burroughs Wellco

and Mark Lavin and E
Bolt Beranek and Ne

Summary

STEREOBOX interactively displays
molecule. The display differs from t
display in that the image for on
STEREOBOX display can be viewed with
containing polarized filters and a ha

Calling Seque

```
RAC PUBLIC STEREOBOX
CALL STEREOBOX
```

Entry

```
CALL STEREOBOXQT(mol-name, big-s
                 <molecule>   <bo

                 screen-center, s
                   <vector>
```

Citation

Cory, M. and White, R. F.;
Molecules (ed. W. P. Rindone and T
Newman Inc., Cambridge, MA; April 19

Questions or Com

Contact Eric Meyer on the PROPHE

Things to Reme

You must have a stereo view
STEREOBOX. The model coordinates
before calling STEREOBOX. STEREOBOX
only.

rt F. White
e Co.

ic Meyer
man Inc.

a stereoscopic image of a
e standard stereo molecule
eye is reversed. The
olarized glasses and a box
f-silvered mirror.

<u>ce</u>

```
reen, vertical,
l>     <bool>

lect-col)
fixed>
 or
text>
```

"STEREOBOX" in <u>PROPHET</u>
Kush); Bolt Beranek and
0.

<u>ents</u>

r line.

<u>ber</u>

g box available to use
f the molecule must be set
vorks on display terminals

STEREOBOX interactively displays two images of the given molecule on the screen, one for each eye. The left eye image is rotated slightly with respect to the right around the vertical axis, so that the images can be fused stereoscopically into an image with depth. The difference between STEREOBOX and the standard PROPHET command "DISPLAY mol-name STEREO" is that STEREOBOX reflects the left image along the horizontal axis. This method of projection is described in Ortony, A. A. (1970) Comp. J. 14:140-144.

The STEREOBOX images cannot, therefore, be fused with the naked eye or standard stereo lenses. To see the stereo image you must wear special glasses and look through a specially designed box. The lenses of the glasses are polarized filters at different orientations. The box contains two polarized filters to direct one image to each eye, and a half-silvered mirror to reflect one image into your direction of view. If your PROPHET site does not already have a stereo viewing box, and you would be interested in getting one, please see the information provided below or call your PROPHET representative for additional information.

Interacting with STEREOBOX

When you call the STEREOBOX procedure, you will be asked a number of questions to set up the display. In order, the questions are:

1. PLEASE ENTER NAME OF MOLECULE? You should respond with the name of your molecule, which must be present in your workarea. It must have model coordinates.

2. ARE YOU WORKING WITH A BIG SCREEN (4014)? You should answer "NO" if you are on a regular PROPHET terminal (Tektronix 4010), and "YES" if you are on a large screen (Tektronix 4014). Most PROPHET terminals are 4010's, so if you are not sure you should just answer in the negative. (This question is necessary because the distance between your eyes is, we hope, a constant, but the distance between two identically-addressed points varies from terminal to terminal. Thus, your mind cannot fuse the two images if they are the wrong distance apart.)

3. DO YOU WANT IMAGES ORIENTED OVER/UNDER (RATHER THAN LEFT/RIGHT)? This simply depends on whether you feel more comfortable holding the stereo box vertically (OVER/UNDER) or horizontally (LEFT/RIGHT).

4. DO YOU WANT TO ENTER A SCREEN OFFSET? The screen offset determines the position of the center of the display: the point midway between the images for the two eyes. If you respond "NO" to this question, the display will be centered on the screen. If you respond "YES", you will be asked:

CENTER X=?
CENTER Y=?

These numbers determine the display center. (Remember: display screen geometry extends approximately from -500 to 500 in X, and -400 to 400 in Y.)

5. DO YOU WISH TO DISPLAY ATOMS SELECTIVELY? If you answer "NO", the whole molecule is displayed. If you answer "YES", you will be asked:

   SELECT ATOMS BY?

   You should respond with the column heading or column number of a BOOLEAN column in the molecule's ATOMS table specifying the desired atoms. An easy way to create this column is to use the PROPHET command "MARK ATOMS OF <mol-name> FOR <subset-name>" and then point to the atoms you wish to include before calling STEREOBOX (see Chapter II, Part 3). You may then respond to this question with the name of the subset you created.

When you have answered these questions, the stereo images will be displayed for you to look at using the box. At this point, you enter an interactive quiz allowing you to move the molecule around at will. There are five commands in STEREOBOX:

1. TURN (T)

   which turns both images around the X, Y, or Z axis. It asks:

   DEGREES?
   AXIS (X, Y, Z)?

   X is left-to-right, Y is bottom-to-top, and Z is perpendicular to the screen, back-to-front.

2. SCALE (S)

   which asks for a

   SCALE FACTOR?

   and expands or shrinks the images by that factor. (Negative scale factors cause a reflection through the origin.)

3. MOVE (M)

   which moves the molecule horizontally and/or vertically by asking:

```
X AMOUNT (-500 TO 500) =?
Y AMOUNT (-400 TO 400) =?
```

   This is useful if the molecule is too large to see on the screen all at once.

4. FREEZE (F)

   which "freezes" the molecule exactly like the standard PROPHET FREEZE command. After a FREEZE, the molecule will continue to be displayed with the orientation and scale factor it had when you last froze it.

5. QUIT (Q)

   which stops STEREOBOX and returns you to top-level PROPHET.

STEREOBOXQT

STEREOBOXQT is a quiet entry to STEREOBOX which bypasses all questions. The molecule is displayed just once, according to the arguments you pass to STEREOBOXQT. You do not enter the interactive quiz described above.

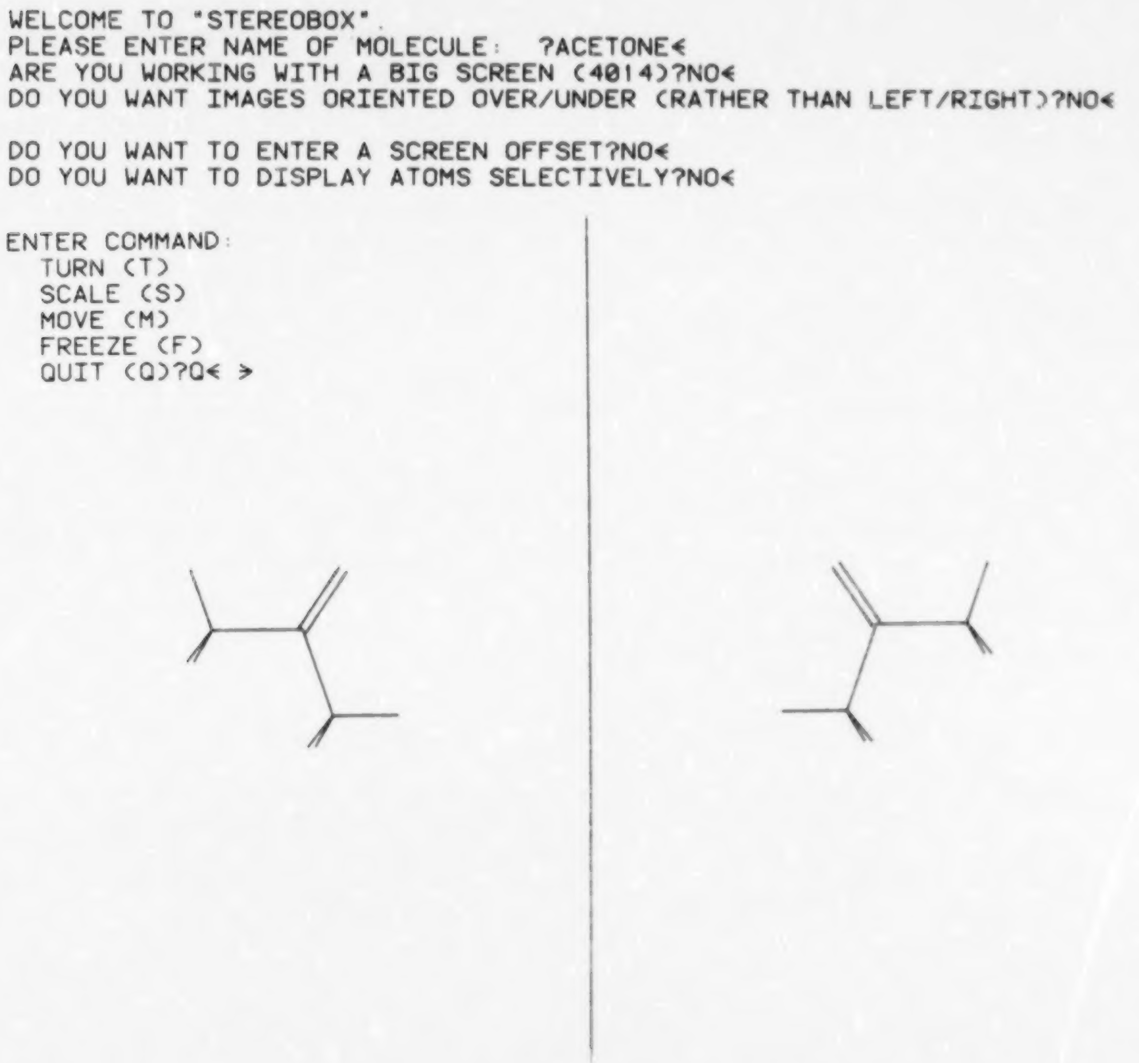
The arguments to STEREOBOXQT have the following meanings:

1. "mol-name" is the molecule to be displayed. It must already have model coordinates.
2. "big-screen" is a BOOL specifying whether or not you are on a big-screen terminal (4014) or not. TRUE means you are on a big screen, FALSE or EMPTY means a small screen (4010).
3. "vertical" is a BOOL specifying how you want the images displayed on the screen. If VERT is TRUE or EMPTY, the image for the right eye will appear below that for the left. If VERT is FALSE, the images will appear side by side.
4. "screen-center" is a VECTOR (a FLOAT ARRAY) which tells where the screen center point is: the point midway between the two images. "screen-center" should specify both the X and Y center coordinate. If this argument is left empty, a default SCRCTR of [0.0, 0.0] is assumed.

5. "select-col" is an optional FIXED or TEXT argument to STEREOBOXQT, allowing you to select only certain atoms of the molecule. If it is EMPTY, all the atoms are shown. Otherwise, it must represent the FIXED column number or TEXT heading of a selecting column in the ATOMS table of the molecule. That column must be of type BOOL, and the only atoms to be displayed will be those whose entry contains TRUE in the selecting column. This column can be conveniently made before calling STEREOBOXQT by using the "MARK ATOMS OF mol-name FOR subset-name" command.

EXAMPLE

Suppose you want to use STEREOBOX on your ACETONE molecule. ACETONE is present in your workarea and has already had its model computed. You can then proceed as follows:



```
WELCOME TO "STEREOBOX".
PLEASE ENTER NAME OF MOLECULE:  ?ACETONE<
ARE YOU WORKING WITH A BIG SCREEN (4014)?NO<
DO YOU WANT IMAGES ORIENTED OVER/UNDER (RATHER THAN LEFT/RIGHT)?NO<

DO YOU WANT TO ENTER A SCREEN OFFSET?NO<
DO YOU WANT TO DISPLAY ATOMS SELECTIVELY?NO<

ENTER COMMAND:
  TURN (T)
  SCALE (S)
  MOVE (M)
  FREEZE (F)
  QUIT (Q)?Q< >
```

Appendix

ORDERING INFORMATION FOR STEREO BOX

The stereo box viewing system for PROPHET is identical to the "AMSOM Box", the viewer sold with the Atlas of Macromolecular Structure on Microfiche (AMSOM). Alternatively, you may use a home built mirror can be used for viewing the whole screen, since the AMSOM box is only 9 x 5 inches.

The AMSOM set is available from:

Tracor Jitco Inc.
1776 East Jefferson Street
Rockville, MD 20852

To build the mirror system a plexiglass frame can be attached to a hood placed around the Tektronix screen. Tektronix sells a viewing hood for the terminals which allows better viewing of the screen in high light environments. The mirror can be ordered from the company listed below. They will cut the mirror to fit the screen if a template is included in the order.

The ordering information for obtaining the materials you need to construct your own AMSOM viewer is listed below:

Mirror: Denton Vacuum
8 Springdale Road
Cherry Hill Industrial Center
Cherry Hill, NJ 08003

Order -- MLBS (multi-level beam splitter)
60% transmittance 40% relecting
size to fit screen (for 4014, it was 11 x 15 -- $100)
Mr. Peter Denton was the contact person.

Filters: Polaroid Corporation
549 Technology Square
Cambridge, MA 02139

HN32 linear polarizer -- standard .030 in. (thick)

Glasses: The AMSOM glasses may be used or glasses may be constructed from the excess Polaroid filter and standard safety glasses without the plastic lenses. Note also that plastic eyeglass lenses tend to depolarize light and blur the images.

TABS2MOL

by Gail M. Rubin
Bolt Beranek and Newman Inc.

Summary

TABS2MOL creates a valid molecule from tables supplied.

Calling Sequence

```
RAC PUBLIC TABS2MOL
CALL TABS2MOL(atoms-table, bonds-table, constraints-table)
                <table>      <table>          <table>
```

Entry

```
CALL TABS2MOLQT(atoms-table, bonds-table, constraints-table,
                  <table>      <table>          <table>
                mol-type, new-molecule)
                 <text>    <molecule>
```

Citation

No citation necessary.

Questions or Comments

Contact Gail Rubin on the PROPHET line.

Things to Remember

The constraints-table argument may be EMPTY. The atoms and bonds tables must have the correct formats, and must represent a fully connected molecule.

TABS2MOL works on all kinds of terminals.

The public procedure TABS2MOL is useful if you have atoms and bonds tables and you want a molecule set up with those tables.

TABS2MOL will ask you if you want the molecule to be disk-based. For normal molecules, answer NO (N). If the molecule is very large (more than 400 atoms), answer YES (Y).

Next, TABS2MOL does some error checking. Some of the possible error messages are:

```
ATOMS TABLE HAS INCORRECT FORMAT, TOO FEW COLUMNS
ATOMS TABLE HAS COLUMN[S] OF INCORRECT TYPE.
```

(These messages can also appear for the bonds and constraints tables.) See the attached figures for the correct structures of the tables.

If the entries in the BOND ARRAY column of the atoms table for some atom A don't point to bonds which have atom A in their ATOM 1 or Atom 2 columns, you will see error messages of this form:

```
ATOM 10 HAS INCORRECT BOND ARRAY.
BOND 5 HAS INCORRECT ATOM CONNECTIONS.
MOLECULE INVALID.
```

If any bonds have EMPTY for the entries in columns 2 and 3 (ATOM 1 and ATOM 2), the error message will be:

```
BOND 9 IS MISSING AN ATOM.
```

Finally, if TABS2MOL determines that the molecule has more than one fully connected piece, it will type out a message with the indices of the atoms that aren't connected to the largest section of the molecule.

If there are no errors found, TABS2MOL will ask for a name that the new molecule should have when you exit from the procedure.

TABS2MOLQT is a non-interactive version of TABS2MOL. The fourth argument must be a TEXT; either 'DISK' or 'CORE'. Use

'CORE' for a normal molecule, use 'DISK' for very large molecules. The last argument, (new molecule) must be declared before calling TABS2MOLQT; it will be set to the molecule created.

(format of required tables)

ATOMS 0R X 7C

	1 TYPE <TEXT>	2 BOND ARRAY <FIXED ARRAY>	3 SKETCH X <FLOAT>	4 SKETCH Y <FLOAT>	5 MODEL X <FLOAT>	6 MODEL Y <FLOAT>	7 MODEL Z <FLOAT>

BONDS 0R X 3C

	1 TYPE <TEXT>	2 ATOM 1 <FIXED>	3 ATOM 2 <FIXED>

CONSTRAINTS 0R X 4C

	1 TYPE <TEXT>	2 ATOMS <FIXED ARRAY>	3 BONDS <FIXED ARRAY>	4 VALUE <ANY>

Appendices and Index

Appendix A

PUBLIC TABLES
CONTAINING CHEMICAL INFORMATION

This appendix presents listings of four PUBLIC PROPHET tables that contain information used by the PROPHET model builder. The first three tables -- ELEMENTS, BONDLENGTHS and DIRECTIONS -- are PUBLIC TABLE copies of the internal information used by the model builder in determining bond lengths and bond angles outside of ring systems. The last table -- PUBLIC RINGS -- is used, together with your project's SHARABLE RINGS table (if it has one) and your own PRIVATE RINGS table (if you have one) to assign coordinates to atoms within ring systems if the same ring system with coordinates is already present in one of the RINGS tables. The way in which the model builder uses this information is described in Chapter II, Part 2.

The tables in this appendix are accurate representations of the public tables at press time, but they are updated with additional information from time to time. In order to see the current version of the ELEMENTS, BONDLENGTHS and DIRECTIONS tables, you must ACCESS the table first. For example, to get a hard copy of the latest version of the ELEMENTS table, use the following commands:

```
ACCESS PUBLIC ELEMENTS<GO>
PRINT ELEMENTS<GO>
```

You do not need to ACCESS the PUBLIC RINGS table; you may display or print it at any time. You cannot change any information in any of these tables yourself. If you have requests for element information or ring structures to be added to these tables, please call your PROPHET representative.

ELEMENTS:

105 ROWS X 5 COLUMNS

0 SYMBOL	1 FULL NAME	2 WEIGHT	3 COVALENT RADIUS	4 VANDERWAALS RADIUS	5 BOND CODES (SEE DIRECTIONS TABLE)
1. C	CARBON	12.011	.77	1.6	[1,2,3,8,14]
2. H	HYDROGEN	1.008	.3	1.	[10]
3. O	OXYGEN	16.	7.	1.4	[4,11]
4. N	NITROGEN	14.008	.72	1.5	[5,6,12,15,13]
5. CL	CHLORINE	35.457	.99	1.8	[10]
6. F	FLUORINE	19.	.68	1.35	[10]
7. BR	BROMINE	79.916	1.14	1.95	[10]
8. I	IODINE	126.904	1.33	2.15	[10]
9. S	SULFUR	32.066	1.04	1.85	[9,4,7,11]
10. P	PHOSPHORUS	30.975	.96	1.8	[7,13,16,17]
11. E	FREE	0.	0.	0.	[10]
12. ?	QUEST	0.	0.	0.	[1,2,3,8,14]
13. HE	HELIUM	4.003			
14. LI	LITHIUM	6.939			[10]
15. BE	BERYLLIUM	9.012			[4,11]
16. B	BORON	10.811			[5,6,12,15]
17. NE	NEON	20.183			
18. NA	SODIUM	22.99			[10]
19. MG	MAGNESIUM	24.312			[4,11]
20. AL	ALUMINUM	26.982			[5,6,12,15]
21. SI	SILICON	28.086			[1,2,3,8,14]
22. AR	ARGON	39.948			
23. K	POTASSIUM	39.102			[10]
24. CA	CALCIUM	40.08			[4,11]
25. SC	SCANDIUM	44.956			[5,6,12,15]
26. TI	TITANIUM	47.9			[5,6,12,15]
27. V	VANADIUM	50.942			[5,6,12,15]
28. CR	CHROMIUM	51.996			[9]
29. MN	MANGANESE	54.938			[1,2,3,8,14]
30. FE	IRON	55.847			[5,6,12,15]
31. CO	COBALT	58.933			[5,6,12,15]
32. NI	NICKEL	58.71			[4,11]
33. CU	COPPER	63.54			[4,11]
34. ZN	ZINC	65.37			[4,11]
35. GA	GALLIUM	69.72			[5,6,12,15]
36. GE	GERMANIUM	72.59			[1,2,3,8,14]
37. AS	ARSENIC	74.922			[5,6,12,15]
38. SE	SELENIUM	78.96			[4,11]
39. KR	KRYPTON	83.8			
40. RB	RUBIDIUM	85.47			[10]

ELEMENTS:

105 ROWS X 5 COLUMNS

0 SYMBOL	1 FULL NAME	2 WEIGHT	3 COVALENT RADIUS	4 VANDERWAALS RADIUS	5 BOND CODES (SEE DIRECTIONS TABLE)
41. SR	STRONTIUM	87.62			[4,11]
42. Y	YTTRIUM	88.905			[5,6,12,15]
43. ZR	ZIRCONIUM	91.22			[1,2,3,8,14]
44. NB	NIOBIUM	92.906			[5,6,12,15]
45. MO	MOLYBDENUM	95.94			[9]
46. TC	TECHNETIUM	99.			[9]
47. RU	RUTHENIUM	101.07			[5,6,12,15]
48. RH	RHODIUM	102.905			[5,6,12,15]
49. PD	PALLADIUM	106.4			[4,11]
50. AG	SILVER	107.87			[10]
51. CD	CADMIUM	112.4			[4,11]
52. IN	INDIUM	114.82			[5,6,12,15]
53. SN	TIN	118.69			[4,11]
54. SB	ANTIMONY	121.75			[5,6,12,15]
55. TE	TELLURIUM	127.6			[4,11]
56. N+	QUATERNARY-N	14.008	.73	1.5	[1,2,3,8,14]
57. XE	XENON	131.3			
58. CS	CESIUM	132.905			[10]
59. BA	BARIUM	137.34			[4,11]
60. LA	LANTHANUM	138.91			[5,6,12,15]
61. CE	CERIUM	140.12			[5,6,12,15]
62. PR	PRASEODYMIUM	140.907			[5,6,12,15]
63. ND	NEODYMIUM	144.24			[5,6,12,15]
64. PM	PROMETHIUM	147.			[5,6,12,15]
65. SM	SAMARIUM	150.35			[4,11]
66. EU	EUROPIUM	151.96			[4,11]
67. GD	GADOLINIUM	157.25			[5,6,12,15]
68. TB	TERBIUM	158.93			[5,6,12,15]
69. DY	DYSPROSIUM	162.5			[5,6,12,15]
70. HO	HOLMIUM	164.93			[5,6,12,15]
71. ER	ERBIUM	167.26			[5,6,12,15]
72. TM	THULIUM	168.934			[5,6,12,15]
73. YB	YTTERBIUM	173.04			[4,11]
74. LU	LUTETIUM	174.97			[5,6,12,15]
75. HF	HAFNIUM	178.49			[1,2,3,8,14]
76. TA	TANTALUM	180.948			[7,13,16,17]
77. W	TUNGSTEN	183.85			[9]
78. RE	RHENIUM	186.2			
79. OS	OSMIUM	190.2			[5,6,12,15]
80. IR	IRIDIUM	192.2			[5,6,12,15]

ELEMENTS:

105 ROWS X 5 COLUMNS

	0 SYMBOL	1 FULL NAME	2 WEIGHT	3 COVALENT RADIUS	4 VANDERWAALS RADIUS	5 BOND CODES (SEE DIRECTIONS TABLE)
81.	PT	PLATINUM	195.09			[4,11]
82.	AU	GOLD	196.967			[10]
83.	HG	MERCURY	200.59			[4,11]
84.	TL	THALLIUM	204.37			[10]
85.	PB	LEAD	207.19			[4,11]
86.	BI	BISMUTH	208.98			[5,6,12,15]
87.	PO	POLONIUM	210.			
88.	AT	ASTATINE	210.			[10]
89.	RN	RADON	222.			
90.	FR	FRANCIUM	223.			[10]
91.	RA	RADIUM	226.			[4,11]
92.	AC	ACTINIUM	0.			
93.	TH	THORIUM	232.038			[1,2,3,8,14]
94.	PA	PROTACTINIUM	231.			
95.	U	URANIUM	238.03			[1,2,3,8,14]
96.	NP	NEPTUNIUM	237.			[1,2,3,8,14]
97.	PU	PLUTONIUM	242.			[5,6,12,15]
98.	AM	AMERICIUM	243.			[5,6,12,15]
99.	CM	CURIUM	247.			[5,6,12,15]
100.	BK	BERKELIUM	249.			[5,6,12,15]
101.	CF	CALIFORNIUM	251.			
102.	ES	EINSTEINIUM	254.			
103.	FM	FERMIUM	253.			
104.	MD	MENDELEVIUM	256.			
105.	NO	NOBELIUM	254.			

```
BONDLENGTHS:  BOND LENGTHS USED BY PROPHET MODEL BUILDER

22 ROWS  X  4 COLUMNS

     1      2      3    4
     ATOM 1 ATOM 2 BOND BOND
                   TYPE LENGTH
  -----------------------------
   1 C      C      S    1.54
   2 C      C      D    1.34
   3 C      C      T    1.2
   4 C      C      R    1.39
   5 C      H      S    1.1
  -----------------------------
   6 C      N      S    1.49
   7 C      N      D    1.27
   8 C      N      R    1.33
   9 C      O      S    1.43
  10 C      O      D    1.23
  -----------------------------
  11 C      S      S    1.81
  12 C      S      D    1.61
  13 C      S      T    1.47
  14 C      CL     S    1.7
  15 BR     C      S    1.91
  -----------------------------
  16 C      F      S    1.4
  17 C      I      S    2.11
  18 H      N      S    1.02
  19 N      O      S    1.4
  20 N      O      D    1.24
  -----------------------------
  21 C      N+     S    1.49
  22 H      N+     S    1.03
```

```
DIRECTIONS:

17 ROWS  X  2 COLUMNS


   1            2
   BOND PATTERN DIRECTION VECTORS
-------------------------------------------------------------
 1 [S,S,S,S]    [[-1.,0.,0.],[.33381,.94264,0.],[.33381,-.47-
                132,-.81635],[.33381,-.47132,.81635]]
 2 [S,S,D]      [[-1.,0.,0.],[.33381,.94264,0.],[.55919,-.82-
                904,0.]]
 3 [S,T]        [[-1.,0.,0.],[1.,0.,0.]]

 4 [S,S]        [[-1.,0.,0.],[.30902,.95106,0.]]

 5 [S,S,S]      [[-1.,0.,0.],[.30902,.95106,0.],[.30902,-.42-
                535,.85064]]
-------------------------------------------------------------
 6 [S,D]        [[-1.,0.,0.],[.45399,.89101,0.]]

 7 [S,S,S,D]    [[-1.,0.,0.],[.33381,.94264,0.],[.33381,-.47-
                132,-.81635],[.33381,-.47132,.81635]]
 8 [D,D]        [[-1.,0.,0.],[1.,0.,0.]]

 9 [S,S,D,D]    [[-1.,0.,0.],[.33381,.94264,0.],[.33381,-.47-
                132,-.81635],[.33381,-.47132,.81635]]
10 [S]          [[-1.,0.,0.]]
-------------------------------------------------------------
11 [D]          [[-1.,0.,0.]]

12 [T]          [[-1.,0.,0.]]

13 [D,D,S]      [[-1.,0.,0.],[.5,.866025,0.],[.5,-.866025,0.-
                ]]
14 [R,R,S]

15 [R,R]
-------------------------------------------------------------
16 [S,S,T]

17 [D,R,R]
```

PUBLIC RINGS:

13 ROWS X 6 COLUMNS

0 NAME	1 ENTRY	3 REFERENCE
1. BENZENE	C6 ?6	
2. CYCLOHEXANE	C6 ?12	
3. CYTOSINE	C4 N2 ?5	J. DONAHUE, ARCH. BIOCHEM. BIOPHYS. 128,591(1968).
4. THYMINE	C4 N2 ?6	J. DONAHUE, ARCH. BIOCHEM. BIOPHYS. 128,591(1968).
5. BENZENE (NON-RESONANT FORM)	C6 ?6	
6. RIBOSE	C4 O ?8	A.R.I MUNNS _P. TOLLIN, ACTA.-CRYST. B26,1101(1970).
7. CYTOSINE-THYMINE-(RESONANT FORM)	C4 N2 ?4	J. DONAHUE, ARCH. BIOCHEM. BIOPHYS. 128,591(1968).
8. ADENINE	C5 N4 ?4	J. DONAHUE, ARCH. BIOCHEM. BIOPHYS. 128,591(1968).
9. GUANINE	C5 N4 ?5	J. DONAHUE, ARCH. BIOCHEM. BIOPHYS. 128,591(1968).
10. IMIDAZOLE	C3 N2 ?4	CONSTRUCTED FROM STANDARD BOND-LENGTHS AND ANGLES
11. INDOLE	C8 N ?7	CONSTRUCTED FROM STANDARD BOND-LENGTHS AND ANGLES
12. PROLINE	C4 N ?9	AVERAGED FROM 6 CRYSTALLOGRAPHIC DETERMINATIONS
13. PROLINE+	C4 N+ ?10	AVERAGED FROM 6 CRYTALLOGRAPHIC DETERMINATIONS

Appendix B

AN INTRODUCTION TO THE MEDICAL FOUNDATION OF BUFFALO PUBLIC MOLECULE TABLES AND PROCEDURES

PROPHET molecules created from crystallographically determined atomic coordinates are stored in public tables for several hundred steroids, aromatic amino acids, thyroactive compounds, and prostaglandins. The public table MOLDIRECTORY (Figure 1) is a master directory for the molecule tables and contains the table names, the MFB registry numbers of the specific molecules stored in the tables, and a description of the general class of compound.

All MFB public molecule tables are archived and protected. Thus, they must be dearchived prior to their use. Once dearchived, a SHADOW table is created in your disk area which contains the same data as the archived table. This copy will be deleted after you LOGOUT. You can therefore not dearchive these tables by batch dearchiving. If you want your own copy, you must use the MAKE...FROM...FOR... command.

The saved table structure of the molecule tables includes the MFB registry numbers as row names and the actual molecules stored in column 2. A copy of a molecule may be obtained by the commands

```
DEARCHIVE PUBLIC table-name

RESTORE registry-number FROM table-name [AS mol-name]
```

A list of the MFB registry numbers of the molecules contained in each table may be obtained by saying

```
EAD table-name COL 0
```

Crystallographic molecules contain certain information not found in normal PROPHET molecules. For a crystallographic molecule M, the following table summarizes the information contained in M, and how to access that information. The various fields may be set to new values by using the SET counterpart to each access function (i.e., %SETATTRIBUTE (M,attribute,value), or %SETCELL (M.ATOM,I,col,value)).

Information	Data Type	How to Access
STRUCTURE ID	TEXT	M.NAMES[1]
IUPAC name	TEXT	M.NAMES[2]
Crystallographic cell constants	VECTOR	*%GETATTRIBUTE(M,'CELLCONSTANTS')
Space group	TEXT	%GETATTRIBUTE(M,'SPACEGROUP')
Literature reference	TEXT	%GETATTRIBUTE(M,'REFERENCE')
Name of Ith atom	TEXT	%GETATOMNAME(M,I)
Standard deviation of X coordinate of Ith atom	FLOAT	%GETCELL(M.ATOMS,I,'SDX')
Standard deviation of Y coordinate of Ith atom	FLOAT	%GETCELL(M.ATOMS,I,'SDY')
Standard deviation of Z coordinate of Ith atom	FLOAT	%GETCELL(M.ATOMS,I,'SDZ')

---

*returns a vector of form: [A,B,C,α,β,γ]

In order to convert crystallographic coordinates into cartesian coordinates, you should make use of the procedure %CRYSTXFM. Given the cell constants V (V is a vector of form [A,B,C, , ,]), %CRYSTXFM returns a transformer which converts crystallographic to cartesian coordinates.

The following example demonstrates how crystallographic coordinates may be converted to cartesian coordinates.

Suppose CRYSTCOOR is a VECTOR ARRAY which contains the crystallographic coordinates of several atoms of molecule M. The following piece of code converts these to cartesian coordinates and stores the results in a corresponding VECTOR ARRAY called CART.

```
DECLARE CRYST2CART MATRIX;
CRYST2CART = %CRYSTXFM(%GETATTRIBUTE(M,'CELLCONSTANTS'));
DO I = 1 TO TOP(CRYSTCOOR);

CART[I] = (CRYSTCOOR[I]) XFM CRYST2CART;

END;
```

Atom names are stored in column zero of the atoms table. To allow easy manipulation of atom names, several gatekeeper procedures have been created. These are described in Chapter III, Part 2.

Procedures written at the Medical Foundation for manipulating crystallographic molecules are listed below and are described in Chapter V.

Procedure Name	Purpose
SETMOLNAMES	Set the names subfield of a molecule.
SETATOMNAMES	Name atoms in a molecule.
ORMOL	Orient a molecule by specifying an origin atom, an atom on the X-axis, and two atoms defining a vector parallel to the XY-plane (i.e., plane of the screen).
PLANE	Prepare a PROPHET table containing the distances of all atoms to a least squares plane passed through a specified set of atoms.
CONNLIST	List the connection table for a crystallographic molecule.
DISTAB	Prepare a PROPHET table containing specified distances and angles for several molecules.
MAST	List all bonded distances, angles, and torsion angles in a molecule.
MODIFY	Change the existing molecule by adding or deleting atoms or bonds.
CONNECT	Create a PROPHET molecule from a crystallographic coordinate file or from arrays containing atom coordinates.
FITMOL	Fit a fragment molecule to a parent molecule by minimizing the distances between sets of atoms in each molecule.
GENMOL	Link a fragment molecule to a parent molecule.

Note that several of the MFB molecule procedures (viz., MODIFY, PLANE, MAST, CONNECT) work only on molecules in the MFB molecule tables, or on PROPHET molecules that have had their subfields filled with the requisite information. For example, some of the procedures assume that atoms are assigned names (e.g. C1, C2, etc.). The public procedure SETATOMNAMES can be used to give atoms such names. Some of the MFB molecule procedures also assume that the names subfield of a molecule contains a compound's structure ID and its IUPAC name or at least some text. The public procedure SETMOLNAMES can be used to set these names.

Reference

Further information concerning the steroid molecules is contained in the Atlas of Steroid Structure, edited by W. L. Duax and D. A. Norton, Plenum Press, New York (1975).

MOLDIRECTORY 36R X 3C

	TABLENAME	REGISTRY NUMBERS	DESCRIPTION
1	ANDR1	AN01-AN39	ANDROSTANES
2	ANDR2	AN40-AN69	ANDROSTANES
3	ANDR3	AN70-AN100	ANDROSTANES
4	ANDR4	AN101-AN112	ANDROSTANES
5	ANTF1	AB000-AF2H	ANTIFOLATES
6	BIOT1	BV1A-BV6	BIOTINS
7	CARD1	CR01-CR08	CARDANOLIDES
8	CHOL1	CAN01-CAN11	CHOLANES
9	CST1	CST01-CST24A2*	CHOLESTANES
10	CST2	CST25-CST36A2	CHOLESTANES
11	CYTOCHAL	CYTA-CELO	CYTOCHALASINS
12	DCTHY1	DC01-DC26	DIPHENYLMETHYLS
13	DNTHY1	DNA01-DN19	DIPHENYLAMINES
14	DOTHY1	DO01-DO13	DIPHENYLETHERS
15	DSTHY1	DS01-DS06B	DIPHENYLSULFIDES
16	ERGO1	ER01-ER22	ERGOSTANES
17	ESTR1	ES01-ES30	ESTRANES
18	ESTR2	ES31-ES56	ESTRANES
19	ESTR3	ES57-ES72	ESTRANES
20	ESTR4	ES73-ES85	ESTRANES
21	LAN1	LA01-LA15	LANOSTANES
22	MISTER1	ST01	MISC STEROIDS
23	PG1	PGA1M-PGB1PR	PROSTAGLANDINS
24	PHAL1	P01-P16	PHENYLALANINES
25	PREG1	PR01-PR34	PREGNANES
26	PREG2	PR35-PR64A2	PREGNANES
27	PREG3	PR65-PR94	PREGNANES
28	PREG4	PR95-PR115	PREGNANES
29	PREG5	PR116-PR132	PREGNANES
30	SALK1	SA01-SA14	STEROID ALKALOID
31	SPIR1	SP01-SP04	SPIROSTANS
32	SPNDL1	EPIPODO-BRPODO2	SPINDLE POISONS
33	SYNEST1	SE01-SE13	SYN ESTROGENS
34	THYR1	TO01-TO15	THYRONINES
35	TYRO1	TA01-T24	TYROSINES
36	TYRO2	T25-TB30	TYROSINES

Figure 1. Public Table MOLDIRECTORY

Appendix C

PROPHET CITATIONS

If you are publishing scientific work in which any of the facilities described in this manual played a significant role, you are encouraged to include with your article references including the following citation:

> Rindone, W. P. and Kush, T., ed. PROPHET Molecules: A User's Guide to the Molecule Facilities of the PROPHET System. Bolt Beranek and Newman Inc. Cambridge, MA, April 1980.

We particularly encourage you to include citations to the public procedures described in Chapter V when they have been important to your work. The summary box for each procedure includes appropriate citation information. The information that is appropriate in a citation includes the author(s) of the procedure, the editors and title of this manual, the edition date and the publisher of the manual. The exact format of the citation will depend on the editorial standards set by the publication in which it appears; one typical example is given here.

> Potter, Stephen and Weeks, Charles. ORMOL. In W. P. Rindone and T. Kush, ed. PROPHET Molecules: A User's Guide to the Molecule Facilities of the PROPHET System. Bolt Beranek and Newman Inc. Cambridge, MA, April 1980.

You are also encouraged to indicate the general role that the PROPHET system has played in your work. This may be accomplished through citations, acknowledgements or a combination of the two. The following publications are appropriate for citation.

> Raub, W.F. (1974) "The PROPHET System and resource sharing." Federation Proceedings 33:2390-2392.

> Hollister, C. and Wood, J. J., ed. User's Manual for the PROPHET System. Bolt Beranek and Newman Inc. Cambridge, MA, October 1978.

In any published acknowledgements of the role the PROPHET system has played in your work, it is appropriate for you to cite the fact that the development and maintenance of PROPHET has been funded by the Division of Research Resources, National Institutes of Health.

INDEX

Page
Accessing molecules 3-15
ADD RING command 2-23
ADD RING quiz 2-22, 2-24, 2-28
%ADDATOM example 3-23
%ADDATOM procedure 3-16, 3-18
%ADDBOND example 3-23
%ADDBOND procedure 3-23, 3-24
%ADDCONSTRAINT procedure 3-26, 3-27
Adding atoms 3-18, 3-62, 5-38
Adding bonds 3-23, 5-61
Adding constraints 3-27
Adding molecules to multimolecules 2-63, 3-88
Addition, matrices 4-5, 4-6
Addition, vectors 4-5, 4-6
%ADDNEWATOM example 3-63
%ADDNEWATOM function 3-16, 3-18
%ADIHEDRAL function 3-47, 3-50
Adjusting molecules 3-38
Adjusting multimolecules 3-91
%ADJUSTMOL procedure 3-38, 3-18, 3-21, 3-22, 3-32, 3-39, 3-65, 3-77, 3-88, 3-89
%ADJUSTMULTIMOL procedure 3-86, 3-88, 3-89, 3-91, 3-95
Amide bonds 5-95
Amino acids 5-9, 5-54.1, 5-102
AMINOACIDS, public table 5-54.2*
ANGLE command expression 2-56
Angle determination 4-11
%ANGLE3 function 4-10, 4-11
%ANGLE4 function 4-10, 4-11
Angles, torsion 4-11
Arithmetic operators, matrices 4-5
Arithmetic operators, vectors 4-5
Array constants in vectors and matrices 4-1
Atom index 3-4
Atom information 3-19
Atom name gatekeepers 5-3
Atom names 3-4, 3-19, 5-54.7, 5-91
Atom types 3-4, 3-20, 3-43
Atom types in MAKE MOLECULE 2-7, 2-9
%ATOMNAME2INDEX function 3-16, 3-20
%ATOMNAMES2INDICES function 3-16, 3-20
%ATOMNEIGHBORS function 3-33, 3-34
ATOMS subfield 3-4, 3-2
ATOMS table 3-4, 2-45, 2-48, 2-75, 3-2, 3-5, 3-15, 3-40, 3-42, 3-46, 3-52, 3-64, 3-78, 5-11, 5-114
ATOMS table gatekeepers 3-15
ATOMS table structure 3-5
Atoms, adding 3-62, 5-38
%ATOMSDIST function 3-47, 3-49
Attribute names 3-11
ATTRIBUTES gatekeepers 3-30

* indicates an illustration

Page

ATTRIBUTES of a molecule 2-17
Attributes of molecules B-2
ATTRIBUTES subfield 3-11, 3-2, 3-41, 3-77, 3-84
%AUTOBAT procedure 5-54.6, 5-89
Axes, display 2-50, 2-69

Balancing hydrogens 2-13, 3-41, 3-54, 3-62, 5-39
%BDIHEDRAL function 3-47, 3-50
Best fit plane 5-5
BESTPLANE public procedure 5-3
BIND command 2-77
Binding disk molecules 2-76
BOND ANGLE command expression 2-56
Bond angle constraints 2-33, 2-34, 3-69
Bond angles 2-56, 3-49
BOND ANGLES constraint type 3-10
Bond angles, standard 2-21
Bond direction constraints 2-31
Bond index 3-6
Bond information 3-25, 3-45
BOND LENGTH command expression 2-56
Bond length constraints 2-29, 3-66
Bond lengths 2-56, 3-49
Bond lengths, standard 2-21
BOND light button in MAKE MOLECULE 2-8
Bond types 2-9, 3-6, 3-25, 3-56
%BONDBETWEEN function 3-32, 3-33
%BONDEDCOUNT function 3-47, 3-51
Bonding patterns 2-13
BONDLENGTH constraint type 3-10
BONDLENGTHS public table 3-45, A-5
Bonds in MAKE MOLECULE 2-4, 2-9
BONDS subfield 3-6, 3-2
BONDS table 3-6, 2-45, 2-48, 2-75, 3-2, 3-7, 3-40
3-42, 3-46, 3-52, 3-64, 3-78, 5-114
BONDS table gatekeepers 3-23
BONDS table structure 3-7
Bonds, adding 5-61
Bonds, amide 5-95
Bonds, double 3-6
Bonds, peptide 5-9, 5-95
Bonds, resonant 2-10, 2-26, 3-6, 3-51, 3-56
Bonds, single 3-6
Bonds, triple 3-6
Building transformers 4-16

CCHEMTEXT function 3-74, 3-75
CENTER command 2-54
%CHEMCTNS function 3-72, 3-73
CHEMCTNS operator 2-18
Chemical formulas 2-81
Chemical formulas, empirical 3-74
Chemical formulas, nonstandard 3-77
Chemical formulas, sorting 2-82
Chemical information in public tables A-1

* indicates an illustration

Page
3-74, 2-16, 2-81, 3-12
CHEMTEXT data type 3-77
CHEMTEXT displays 3-77
CHEMTEXT sorting keys 3-10
CHIRALITY constraint type 2-36, 3-70
Chirality constraints 5-2, C-1
Citations 2-10, 2-25, 2-27, 5-34
CLEAN light button 5-3
CLEAVEMOL public procedure 3-4, 3-6, 3-41
Column heading, WIPKE 4-13
Combining transformers 5-9*
Command example, DISPLAY 2-48
Command expression LABELING 2-56
Command expression, ANGLE 2-56
Command expression, BOND ANGLE 2-56
Command expression, BOND LENGTH 2-55
Command expression, COORDINATES 2-57
Command expression, DIHEDRAL ANGLE 2-55
Command expression, DISTANCE 3-81
Command expression, LABELING 2-54
Command expression, NUMBER 2-14, 2-44
Command expression, SELECTING 2-55
Command expression, XCOORDINATE 2-55
Command expression, YCOORDINATE 2-55
Command expression, ZCOORDINATE 2-75, 2-79
Command qualifier, CORE 2-75
Command qualifier, DISK 2-72
Command qualifier, NOBACKUP 3-93
Command tokens 2-23
Command, ADD RING 2-77
Command, BIND 2-54
Command, CENTER 2-21, 1-2, 2-28, 3-4, 5-9
Command, COMPUTE MODEL . . . .
5-54.6, 5-57, 5-95
3-8
Command, CONSTRAIN 2-33
Command, CONSTRAIN BOND ANGLES 2-29
Command, CONSTRAIN BOND LENGTH 2-36, 2-37
Command, CONSTRAIN CHIRALITY 2-30
Command, CONSTRAIN DIHEDRAL ANGLE 2-31
Command, CONSTRAIN DIRECTIONS 2-54, 2-73
Command, DEEPFREEZE 2-43
Command, DEFAULT MOLECULES 2-78
Command, DELETE 2-78
Command, DELETE SAVED 2-41, 2-73
Command, DISPLAY 2-17
Command, DISPLAY ... FOR 2-48, 5-54.2
Command, DISPLAY ... LABELING 2-44, 3-78
Command, DISPLAY ... SELECTING 3-4
Command, DISPLAY ATOMS 3-11
Command, DISPLAY ATTRIBUTES 3-6
Command, DISPLAY BONDS 3-8
Command, DISPLAY CONSTRAINTS 3-12
Command, DISPLAY FORMULA 3-12
Command, DISPLAY NAMES 3-12
Command, DISPLAY NOTES 2-22
Command, DISPLAY RINGS . . . .

* indicates an illustration

Page
Command, DISPLAY %TRANS 3-12
Command, EAD 2-41
Command, EDIT molecule 2-13
Command, FILL 2-82
Command, FREEZE 2-53, 2-59, 2-73
Command, MAKE CHEMTEXT 2-82, 3-48
Command, MAKE MOLECULE 1-1, 2-3, 2-18, 2-75
Command, MAKE MOLECULE ... FROM 2-14
Command, MAKE MOLECULE ... FROM ... SELECTING 2-15, 3-42
Command, MAKE MULTIMOLECULE 2-61, 2-79, 3-93
Command, MAKE TABLE ... FROM ... FOR 2-17, 2-83
Command, MANIPULATE 2-61, 2-72, 3-93
Command, MARK 3-64, 3-78
Command, MARK ATOMS 2-14, 2-44
Command, MARK BONDS 2-14, 2-44
Command, MOVE 2-49, 2-73
Command, ORIENT 2-50
Command, PRINT 2-41
Command, PROTECT 2-78
Command, REFINE MODEL 2-28
Command, RESTORE 2-14, 2-77
Command, SAVE 2-14, 2-71, 2-77
Command, SCALE 2-48, 2-73
Command, SET 2-57, 2-76, 2-82, 3-13
Command, SORT 2-82
Command, SUPERDELETE 2-78
Command, SUPERDELETE SAVED 2-78
Command, TURN 2-49, 2-73
Command, TWIST 2-53, 2-73
Command, UNBIND 2-76
Command, UNMARK 2-45, 3-78
Commands, molecule 2-1, 1-2
Complex data types 3-3
%COMPOSITION function 3-46, 3-47
Composition of a molecule 2-16, 3-46
COMPUTE MODEL command 2-21, 1-2, 2-28, 3-4, 5-9
5-54.6, 5-57, 5-95
COMPUTE MODEL command example 5-101
%COMPUTEATOMSANGLE function 3-47, 3-49
Conformations of molecules 1-2, 2-21
CONNECT public procedure 5-13, 3-54, 5-22
Connecting atoms 3-54
Connection table of a molecule 3-56, 5-19
Connection tables 3-51, 3-52
%CONNECTMOL example 3-57*
%CONNECTMOL procedure 3-54, 3-55, 5-17, 5-22, 5-70, 5-85
%CONNECTMOL quiz 3-56
CONNLIST public procedure 5-19
%CONNT2A procedure 3-47, 3-51
Constant, %IDENTITYXFM 4-14, 4-18
CONSTRAIN BOND ANGLES command 2-33
CONSTRAIN BOND LENGTH command 2-29
CONSTRAIN CHIRALITY command 2-36, 2-37
CONSTRAIN command 3-8
CONSTRAIN commands, FREE qualifier 2-29, 2-30, 2-32, 2-35, 2-36

* indicates an illustration

Page

CONSTRAIN DIHEDRAL ANGLE command 2-30
CONSTRAIN DIRECTIONS command 2-31
Constraint examples 2-38
Constraint index 3-8
Constraint information 3-28
Constraint type, BOND ANGLES 3-10
Constraint type, BONDLENGTH 3-10
Constraint type, CHIRALITY 3-10
Constraint type, DIRECTIONS 3-10
Constraint type, TORSION 3-10
Constraint types 3-8
CONSTRAINTS of a molecule 2-37
Constraints of a molecule 2-28, 3-66, 5-95
CONSTRAINTS subfield 3-8, 3-2
CONSTRAINTS table 3-8, 2-75, 3-2, 3-9, 3-42, 5-114
CONSTRAINTS table example 5-98, 5-100
CONSTRAINTS table gatekeepers 3-26
CONSTRAINTS table structure 3-9
Constraints, bond angle 2-33, 2-34, 3-69
Constraints, bond directions 2-31
Constraints, bond length 2-29, 3-66
Constraints, chirality 2-36, 3-70
Constraints, dihedral angle 2-30, 3-68
Constraints, direction 3-69
Constraints, torsion angle 3-68
Constructing new molecules 5-1
%CONVCHEMTEXT function 3-75, 3-76
%CONVCHEMTEXTQT function 3-75, 3-76
Conversion of crystal coordinate 4-18
%CONVERT2COREMM procedure 3-87, 3-91
%CONVERT2COREMOL procedure 3-39
%CONVERT2DISKMM procedure 3-86, 3-91
%CONVERT2DISKMOL procedure 3-39, 3-42
Converting molecules 3-42
Coordinate files for molecules 5-13
Coordinate systems, multimolecules 3-96, 2-69, 2-70, 3-83
COORDINATES command expression 2-55
Coordinates, display 3-53
Coordinates, model 3-4, 3-18, 3-62, 3-83
Coordinates, molecular models 2-55, 3-21
Coordinates, screen 3-53
Coordinates, sketch 3-4, 3-18, 3-21
Copying molecules 3-40
Copying multimolecules 3-85
CORE command qualifier 2-75, 2-79
Core molecules 2-75, 3-40, 3-42
Core multimolecules 3-88, 3-91
Covalent radius 3-43
CROSS operator 4-6, 4-8
Cross product 4-6, 4-8
CRYST public procedure 3-54, 5-21
Crystal coordinate conversion 4-18
Crystallographic cell constants 2-17, 4-18
CRYSTQT public procedure entry 5-21
%CRYSTXFM function 4-14, 4-18

* indicates an illustration

Page
Data type subfields . . . 3-3
Data type, CHEMTEXT . . . 3-74, 2-16, 2-81, 3-12
Data type, MATRIX . . . 4-1
Data type, MOLECULE . . . 3-3, 1-1, 1-5, 3-1
Data type, MOLSCREEN . . . 2-20, 2-23, 3-72
Data type, MULTIMOLECULE . . . 3-83, 1-4, 2-61, 3-1
Data type, VECTOR . . . 3-18, 4-1
Data types, complex . . . 3-3
%DBOX procedure . . . 3-53
DEEPFREEZE command . . . 2-54, 2-73
DEFAULT MOLECULES command . . . 2-43
DEFAULT MOLECULES example . . . 2-44*
DELETE command . . . 2-78
DELETE light button . . . 2-10, 2-65, 5-40
DELETE SAVED command . . . 2-78
%DELETEATOM procedure . . . 3-16, 3-18
%DELETEATOMS procedure . . . 3-16, 3-18
%DELETEATTRIBUTE procedure . . . 3-30, 3-31, 3-91
%DELETEBOND procedure . . . 3-24, 3-25
%DELETECONSTRAINT procedure . . . 3-26, 3-27
Deleting atoms . . . 3-18
Deleting bonds . . . 3-25
Deleting constraints . . . 3-27
Deleting disk molecules . . . 2-78
Deleting molecules from a multimolecule . . . 3-89
Descriptors of molecules . . . 2-17
%DET function . . . 4-10, 4-11
Determinant of a matrix . . . 4-11
Determination of angles . . . 4-11
Dictionaries, ring . . . 2-21, 2-22, 3-45
DIHEDRAL ANGLE command expression . . . 2-57
Dihedral angle constraints . . . 2-30, 3-68
Dihedral angle sign conventions . . . 4-11*
Dihedral angles . . . 2-30, 2-56, 3-50, 3-63
Dipole moments . . . 2-17
Direction constraints . . . 2-31, 3-69
DIRECTIONS constraint type . . . 3-10
DIRECTIONS public table . . . A-6
Directions represented as vectors . . . 4-13
DISK command qualifier . . . 2-75
Disk molecules . . . 2-75, 3-12, 3-40, 3-42, 5-54.6
Disk molecules, binding . . . 2-76
Disk molecules, deleting . . . 2-78
Disk molecules, protecting . . . 2-78
Disk molecules, saving . . . 2-77
Disk multimolecules . . . 2-78, 3-85, 3-91
DISPLAY ... FOR command . . . 2-17
DISPLAY ... LABELING command . . . 2-48, 5-54.2
DISPLAY ... LABELING command example . . . 5-54.4*
DISPLAY ... SELECTING command . . . 2-44, 3-78
DISPLAY ... SELECTING command example . . . 5-54.5*
DISPLAY ATOMS command . . . 3-4
DISPLAY ATTRIBUTES command . . . 3-11
Display axes . . . 2-50, 2-69
DISPLAY BONDS command . . . 3-6

* indicates an illustration

Page

DISPLAY command 2-41, 1-2
DISPLAY command example 5-9*
DISPLAY command, multimolecules 2-73
DISPLAY CONSTRAINTS command 3-8
Display coordinates 3-53
Display coordinates, multimolecules 2-69, 2-70
Display examples, molecules 2-42*
DISPLAY FORMULA command 3-12
DISPLAY NAMES command 3-12
DISPLAY NOTES command 3-12
Display qualifier, SKETCH 2-41
Display qualifier, STEREO 2-41
Display qualifier, THREED 2-41
Display qualifiers 2-42
Display qualifiers, multimolecules 2-73
DISPLAY RINGS command 2-22
DISPLAY %TRANS command 3-12
Display transformer of a molecule 3-12
Display transformers 2-59, 3-83, 3-84
Displays of CHEMTEXT 3-77
Displays of molecules 2-41, 1-3
Displays of multimolecules 1-4, 2-73
Displays, labeled 2-47, 3-81
Displays, perspective 2-58*
Displays, scaled 2-48
Displays, selective 2-44, 3-4
DISTAB public procedure 5-25
DISTANCE command expression 2-55
DISTANCE light button 2-71, 5-44
Distances 3-49
Distances, molecular models 2-55
Division, matrices 4-7
Division, vectors 4-7
Donating public procedures 5-3
DOT operator 4-6, 4-7
Dot product 4-6, 4-7
Double bonds 3-6
Double bonds in MAKE MOLECULE 2-6
Dreiding model 2-42
%DTEXT procedure 3-53

EAD command 2-41
EDIT molecule command 2-13
Editing molecular models 5-31
EDITMODEL public procedure 3-65, 5-31, 5-89
EDITMODEL quiz 5-32*
Element information 3-43
Element tables 3-43
Element types 2-9
ELEMENTS public table 5-105, A-2
Empirical formulas 2-81, 2-16, 3-48, 3-74
Empirical formulas, sorting 2-82, 3-77
Energy minimization 2-21, 2-28
Energy minimization program, QCFF 2-28, 5-45, 5-89
Energy minimization program, WIPKE 2-25, 3-4, 3-6

* indicates an illustration

Page
Energy minimization, public procedures 2-28
Examining molecules 2-16
Example, %ADDATOM 3-23
Example, %ADDBONDS 3-23
Example, %ADDNEWATOM 3-63
Example, COMPUTE MODEL command 5-101
Example, %CONNECTMOL 3-57*
Example, CONSTRAINTS table 5-98, 5-100
Example, DEFAULT MOLECULES 2-44*
Example, DISPLAY ... LABELING command 5-54.4*
Example, DISPLAY ... SELECTING command 5-54.5*
Example, DISPLAY command 5-9*
Example, %GETNEWMVEC 3-63
Example, MAKE MOLECULE 2-3
Example, MAKE MULTIMOLECULE 2-63
Example, MAKEPOLYPEPTIDE 5-54.3*
Example, %MARKATOMS 5-10
Example, molecule programming 4-25, 5-2
Example, multimolecule programming 3-90, 3-103, 5-48
Example, ORIENT 2-51*
Example, selective display 2-45, 5-12, 5-54.5
Example, %SETATTRIBUTE 3-77
Example, transformer programming 4-15, 4-25
Example, TURN 2-52*
Example, %TYPEMOL 3-48
Example, %VERIFYMOLCONNS 3-52
Examples, constraints 2-38
Examples, matrix programming 4-21
Examples, molecule displays 2-42*
Examples, molecule programming 3-101
Examples, multimolecule programming 3-98
Examples, substructure search 2-19*
Examples, vector programming 4-21

FILL command 2-82
%FINDBONDANGLE function 3-47, 3-49
%FINDBONDLENGTH function 3-44, 3-45
%FINDELEMENT procedure 3-43, 3-44
FITMOL public procedure 5-47
Flag, %M%DSK 3-12
Flag, %M%FRAG 3-12
Flag, %M%MDL 3-12, 3-40
Flag, %M%SKT 3-12, 3-40
Flag, %M%UNBD 3-12
FLAGS gatekeepers 3-32
FLAGS subfield 3-2, 3-12
FORMULA of a molecule 2-16
FORMULA subfield 3-12, 3-2, 3-74
Formulas, empirical 2-81, 2-16, 3-48, 3-74
Formulas, nonstandard 3-77
Formulas, sorting 3-77
Fragment dictionary, AMINOACIDS 5-54.2*
Fragments, amino acid 5-54.7
Fragments, molecular 2-17, 2-18, 3-72, 5-35, 5-47, 5-51
FREE qualifier, CONSTRAIN commands . . . 2-29, 2-30, 2-32, 2-35, 2-36

* indicates an illustration

Page
Free radicals . . . 2-9
FREEZE command . . . 2-53, 2-59, 2-73
FREEZE light button . . . 5-44
Functions . . . 3-1
Functions, matrix . . . 4-9
Functions, vector . . . 4-9

Gatekeeper procedures . . . 3-3, 3-15
Gatekeepers, atom name . . . 5-3
Gatekeepers, ATOMS table . . . 3-15
Gatekeepers, ATTRIBUTES . . . 3-30
Gatekeepers, BONDS table . . . 3-23
Gatekeepers, CONSTRAINTS table . . . 3-26
Gatekeepers, FLAGS . . . 3-32
Gatekeepers, multimolecule . . . 3-85
Gatekeepers, multimolecule quiz . . . 3-94
Gatekeepers, multimolecule transformers . . . 3-94
GENMOL public procedure . . . 5-51
Geometry, three-dimensional calculations . . . 4-13
%GETATOM1 function . . . 3-24, 3-25
%GETATOM2 function . . . 3-24, 3-25
%GETATOMNAME function . . . 3-16, 3-19
%GETATOMTYPE function . . . 3-16, 3-20
%GETATTRIBUTE function . . . 3-30, 3-31, 3-91
%GETBONDA function . . . 3-16, 3-21, 3-34
%GETBONDTYPE function . . . 3-24, 3-25
%GETCHEMTEXTKEY function . . . 3-75, 3-77
%GETCONATOMS function . . . 3-26, 3-28
%GETCONBONDLENGTH function . . . 3-66, 3-67
%GETCONBONDS function . . . 3-26, 3-28
%GETCONCHIRAL function . . . 3-67, 3-70
%GETCONDIRSANGLES function . . . 3-67, 3-69
%GETCONTORSION function . . . 3-67, 3-68
%GETCONTYPE function . . . 3-26, 3-28
%GETCONVALUE function . . . 3-26, 3-29
%GETFLAG function . . . 3-12, 3-32, 3-33
%GETMVEC function . . . 3-17, 3-22
%GETNEWMVEC example . . . 3-63
%GETNEWMVEC function . . . 3-55, 3-62
%GETRINGMOL function . . . 3-44, 3-45
%GETSVEC function . . . 3-16, 3-21

%HBALANCE procedure . . . 3-62, 3-35, 3-55
Hydrogen balancing . . . 2-13, 3-41, 3-54, 3-62, 5-39
Hydrogens, system-added . . . 2-18, 2-41, 2-43, 3-34, 3-62

Identity matrix . . . 4-12
Identity transformer . . . 4-18
%IDENTITYMATRIX function . . . 4-10, 4-12
%IDENTITYXFM constant . . . 4-14, 4-18
Implementation of multimolecules . . . 3-96
Implementation of transformers . . . 4-18
Index, atom . . . 3-4
Index, bond . . . 3-6
Index, constraint . . . 3-8

* indicates an illustration

Page

Index, molecule in a multimolecule 3-88
Inner product 4-6, 4-7
%INVERSE function 4-9, 4-10
INVOKE light button 3-93, 2-72
%INVXFM function 3-100, 4-14, 4-18

Joining molecules 3-64, 5-35

Label columns 3-81
Labeled displays 2-47, 3-81, 3-82
LABELING command expression 2-48, 3-81
%LASTATOM function 3-16, 3-19
%LASTBOND function 3-24, 3-25
%LASTCONSTRAINT function 3-26, 3-28
Light button, CLEAN 2-10, 2-25, 2-27, 5-34
Light button, DELETE 2-10, 2-65, 5-40
Light button, DISTANCE 2-71, 5-44
Light button, FREEZE 5-44
Light button, INVOKE 3-93, 2-72
Light button, LOCATION 2-25
Light button, MOVE 2-10, 2-27, 2-68, 5-35
Light button, MOVEMOL 2-10, 2-25
Light button, ORIENT 2-69
Light button, PRIORITY 2-25
Light button, REFINE 5-41
Light button, RESTORE 2-11
Light button, SCALE 2-11, 2-25, 2-27, 2-68, 5-35
Light button, TURN 2-11, 2-25, 5-34
Light button, TWIST 2-70, 5-41
Light button, VECTOR 2-71
Light buttons 2-3, 2-61, 5-33
LOCATION light button 2-25
Locking molecules in a multimolecule 2-72

MAG function 4-9, 4-10
Magnitude of a vector 4-9
%MAGSQD function 4-9, 4-10
MAKE CHEMTEXT command 2-82, 3-48
MAKE MOLECULE ... FROM ... SELECTING command 2-15, 3-42, 3-78
MAKE MOLECULE ... FROM command 2-14, 2-15
MAKE MOLECULE command 2-3, 1-1, 2-18, 2-75
MAKE MOLECULE command, NOQUIZ qualifier 2-15
MAKE MOLECULE example 2-3
MAKE MOLECULE quiz . . . 2-3, 1-1, 2-4, 2-5, 2-6, 2-7, 2-8, 2-63, 2-75
3-4, 3-12, 3-35, 3-42, 3-62, 5-67, 5-93
MAKE MOLECULE, atom types 2-7, 2-9
MAKE MOLECULE, BOND light button 2-8
MAKE MOLECULE, bonds 2-4, 2-9
MAKE MOLECULE, double bonds 2-6
MAKE MOLECULE, errors 2-12
MAKE MULTIMOLECULE command 2-61, 2-79, 3-93
MAKE MULTIMOLECULE example 2-63
MAKE TABLE ... FROM ... FOR command 2-17, 2-83
%MAKEMOL procedure . . . . . [illegible]

* indicates an illustration

Page
%MAKEMULTIMOL procedure . . . 3-85, 3-86
MAKEPOLYPEPTIDE example . . . 5-54.3*
MAKEPOLYPEPTIDE, public procedure . . . 5-54.1
MAKEPOLYPEPTIDEQT, public procedure entry . . . 5-54.1
Making formulas . . . 3-74
Making molecular models . . . 5-57
Making molecules . . . 2-3, 1-1, 3-38, 5-67, 5-69, 5-85
Making molecules, public procedures . . . 2-16
Making multimolecules . . . 3-85
MANIPULATE command . . . 2-61, 2-72, 3-93
Manipulating molecules . . . 5-1
Manipulating molecules in multimolecules . . . 2-68
Manipulating multimolecules . . . 2-61
MARK ATOMS command . . . 2-44, 2-14
MARK BONDS command . . . 2-44, 2-14
MARK command . . . 2-44, 3-64, 3-78
%MARKATOMS example . . . 5-10
%MARKATOMS procedure . . . 3-40, 3-79, 3-80
%MARKBONDS procedure . . . 3-40, 3-79, 3-81
MAST public procedure . . . 5-55
%MATDIMEN procedure . . . 4-10, 4-12
Matrices . . . 1-5
Matrix addition . . . 4-5, 4-6
Matrix arithmetic . . . 4-5
Matrix arithmetic operators . . . 4-5
MATRIX data type . . . 4-1
Matrix determinant . . . 4-11
Matrix division . . . 4-7
Matrix functions . . . 4-9
Matrix inversion . . . 4-9
Matrix multiplication . . . 4-5, 4-6
Matrix negation . . . 4-5, 4-6
Matrix operator precedence . . . 4-8
Matrix programming . . . 4-1
Matrix programming examples . . . 4-21
Matrix subtraction . . . 4-5, 4-6
Matrix transposition . . . 4-9
Matrix, identity . . . 4-12
Matrix, transformers . . . 4-13
MBLD public procedure . . . 5-57
%M%DSK flag . . . 3-12
Measurements of molecules . . . 5-25, 5-55
Measuring models, public procedures . . . 2-57
%MERGEMOL procedure . . . 3-55, 3-64
MFB public procedures . . . B-3
MFB public tables . . . B-1
MFBADDBONDS public procedure . . . 5-61
%M%FRAG flag . . . 3-12
%MMADDMOL procedure . . . 3-86, 3-88
%MMDELETEMOL procedure . . . 3-86
%M%MDL flag . . . 3-12, 3-40
%MMGETMOL function . . . 3-86, 3-88
%MMGETMVEC function . . . 3-89, 3-90, 3-96, 3-98
%MMGETMXFM function . . . 3-94, 3-97, 3-100

Page
%MMMANIPMOL procedure 3-86, 3-90, 3-98
%MMQMULTIMOL function 3-94, 3-95
%MMQREFRESH procedure 3-94, 3-95
%MMSETMVEC function 3-89
%MMSETMXFM procedure 3-94, 3-97
Model angles 2-56
Model coordinates 2-55, 3-4, 3-18, 3-21, 3-62, 3-83
Model coordinates in multimolecules 3-89
Model distances 2-55
Model space 3-4
Model torsion angles 2-57
Models of molecules 2-21, 1-2, 3-54, 5-57
MODIFY public procedure 5-63, 5-20
Modifying molecules 3-15, 5-1, 5-63
%MOLBONDLENGTH function 3-47, 3-49
MOLCONNECT public procedure entry 5-13, 5-22, 5-85
%MOLCOORDS procedure 3-47, 3-53
%MOLCOPY function 3-39, 3-40, 3-89
MOLEC subfield 3-83
Molecular composition 3-46
Molecular fragments 2-17, 2-18, 3-72, 5-35, 5-47, 5-51
Molecular measurements 5-25, 5-55
Molecular models 3-54, 5-57
Molecular models, editing 5-31
Molecular properties 3-46, 5-2
Molecular weight 3-48
Molecule commands 2-1, 1-2
Molecule constraint types 3-8
MOLECULE data type 3-3, 1-5, 3-1
Molecule data type 1-1
Molecule descriptors 2-17
Molecule display examples 2-42*
Molecule display transformer 2-59, 3-12
Molecule displays 2-41, 1-2, 1-3, 5-2
Molecule displays, public procedures 2-60
Molecule displays, space-filling 5-103
Molecule flag %M%UNBD 3-12
Molecule flag, %M%DSK 3-12
Molecule flag, %M%FRAG 3-12
Molecule flag, %M%MDL 3-12
Molecule flag, %M%SKT 3-12
Molecule functions 3-1
Molecule gatekeeper procedures 3-3, 3-15
Molecule index in a multimolecule 3-88
Molecule names 2-7, 2-11, 2-17, 5-93
Molecule procedures 3-1
Molecule procedures, public 5-1
Molecule programming 3-1, 1-5
Molecule programming example 4-25, 5-2
Molecule programming examples 3-101
Molecule screening 2-18
Molecule subfield, ATOMS 3-4, 3-2
Molecule subfield, ATTRIBUTES 3-11, 3-2, 3-41, 3-77
Molecule subfield, BONDS 3-6, 3-2
Molecule subfield, CONSTRAINTS 3-8, 3-2

* indicates an illustration

Page
3-2, 3-12
Molecule subfield, FLAGS . . . 3-12, 3-2
Molecule subfield, FORMULA . . . 3-12, 3-2, 3-41
Molecule subfield, NAMES . . . 3-12, 3-2, 3-41
Molecule subfield, NOTES . . . 3-12, 3-2, 3-41, 3-83, 3-99
Molecule subfield, %TRANS . . . 3-3
MOLECULE subfields . . . 3-3
MOLECULE tables . . . 2-17
Molecule, ATTRIBUTES of . . . 2-16
Molecule, composition . . . 5-19
Molecule, connection table . . . 2-37
Molecule, CONSTRAINTS of . . . 2-16
Molecule, FORMULA of . . . 2-17
Molecule, NAMES of . . . 2-16
Molecule, PERCOMP of . . . 3-15
Molecules, accessing . . . 3-38
Molecules, adjusting . . . 3-15
Molecules, ATOMS table . . . B-2
Molecules, attributes . . . 2-28, 3-66
Molecules, constraints . . . 3-42
Molecules, converting . . . 3-40
Molecules, copying . . . 2-75, 3-40, 3-42
Molecules, core-based . . . 2-30*
Molecules, dihedral angles . . . 2-75, 1-5, 3-12, 3-40, 3-42
Molecules, disk-based . . . 2-42
Molecules, display qualifiers . . . 2-13
Molecules, editing . . . 2-16
Molecules, examining . . . 2-54
Molecules, examining models . . . 5-35
Molecules, joining . . . 2-47
Molecules, labeled displays . . . 2-3, 1-1, 3-38, 5-67, 5-69, 5-85
Molecules, making . . . 2-56
Molecules, model angles . . . 2-55
Molecules, model coordinates . . . 2-55
Molecules, model distances . . . 2-57
Molecules, model torsion angles . . . 2-21, 1-2
Molecules, models of . . . 3-15, 5-63
Molecules, modifying . . . 5-77
Molecules, orienting . . . 1-2, 1-4, 2-57
Molecules, perspective displays . . . 2-28, 5-41, 5-89
Molecules, refining models of . . . 2-21
Molecules, rings . . . 2-14
Molecules, saving . . . 2-48
Molecules, scaled displays . . . 2-44
Molecules, selective displays . . . 2-3, 2-41
Molecules, sketches . . . 1-1
Molecules, sketching . . . 1-2, 1-3, 5-107
Molecules, stereo displays . . . 2-17
Molecules, structure searches . . . 2-17
Molecules, substructure searches . . . 1-2
Molecules, three-dimensional models of . . . 2-30
Molecules, torsion angles . . . 5-67
MOLECYCLE public procedure . . . 5-69
MOLEDAT public procedure . . . 2-20, 2-23, 3-72
MOLSCREEN data type . . . 3-47, 3-48
%MOLWEIGHT function . . .

* indicates an illustration

best copy available

Page

MOVE command 2-49, 2-73
MOVE light button 2-10, 2-27, 2-68, 5-35
MOVEMOL light button 2-10, 2-25
%M%SKT flag 3-12, 3-40
%MULTIMOLCOPY function 3-85, 3-86
Multimolecule coordinate reference frames 3-83
MULTIMOLECULE data type 3-83, 2-61, 3-1
Multimolecule display qualifiers 2-73
Multimolecule displays 1-4, 2-73
Multimolecule gatekeepers 3-85
Multimolecule implementation 3-96
Multimolecule programming example 3-90, 3-103
Multimolecule programming examples 3-98, 5-48
Multimolecule quiz 2-61, 2-79, 3-93
Multimolecule quiz gatekeepers 3-94
Multimolecule quiz, tablet indications 3-95
Multimolecule reference frame 2-69, 3-89
Multimolecule subfield, ATTRIBUTES 3-84
Multimolecule subfield, MOLEC 3-83
Multimolecule subfield, %SYSCTR 3-84, 3-99
Multimolecule subfields 3-83
Multimolecule transformer gatekeepers 3-94
Multimolecule transformers 3-99*
Multimolecule, adding molecules 3-88
Multimolecule, coordinate systems 3-96*
Multimolecule, deleting molecules 3-89
Multimolecule, molecule index 3-88
Multimolecules 2-61, 3-83
Multimolecules, adding molecules 2-63
Multimolecules, adjusting 3-91
Multimolecules, coordinate systems 2-69, 2-70
Multimolecules, copying 3-85
Multimolecules, core-based 3-88, 3-91
Multimolecules, disk-based 2-78, 3-85, 3-91
Multimolecules, DISPLAY command 2-73
Multimolecules, display coordinates 2-69, 2-70
Multimolecules, locking molecules 2-72
Multimolecules, making 3-85
Multimolecules, manipulating 2-61
Multimolecules, manipulating component molecules 2-68
Multimolecules, perspective displays 2-73
Multimolecules, setting model coordinates 3-89
Multiplication, matrices 4-5, 4-6
Multiplication, vectors 4-5, 4-6
%M%UNBD flag 3-12

NAMES of a molecule 2-17
Names of atoms 3-4, 3-19
Names of molecules 2-7, 2-11, 2-17, 5-93
NAMES subfield 3-12, 3-2, 3-13, 3-41
Newman projection 2-30, 3-50, 4-11
Newman projections 5-73*
NEWMAN public procedure 5-73
Nitrogen, quaternary 2-9
NOBACKUP command qualifier 2-72

* indicates an illustration

Page
Nonstandard formulas 3-77
NOQUIZ, MAKE MOLECULE command qualifier 2-15
NOTES subfield 3-12, 3-2, 3-41
NUMBER command expression 2-54

Omega torsion angles in polypeptides 5-54.6, 5-97
Operator precedence, matrices 4-8
Operator precedence, vectors 4-8
Operator, CHEMCTNS 2-18
Operator, CROSS 4-6, 4-8
Operator, DOT 4-6, 4-7
Operator, XFM 4-13, 4-14, 4-19
Operators, matrices 4-5
Operators, vectors 4-5
ORIENT command 2-50
ORIENT example 2-51*
ORIENT light button 2-69
Orienting molecules 5-77
ORMOL public procedure 5-77
ORTEP2 program 2-58
%OTHERATOM function 3-33, 3-34

Peptide bonds 5-9, 5-95
PERCOMP of a molecule 2-16
Perpendicular vectors 4-9
%PERPTO function 4-9, 4-10
Perspective displays 2-57*
Perspective displays of molecules 1-2, 1-4
Perspective displays of multimolecules 2-73
%PERSPECTIVE procedure 2-58, 2-73
%PERSPECTIVE 5-104
%PERSPECTIVEQK procedure 2-60, 1-2
Phi torsion angles in polypeptides 5-54.1, 5-97
PHIPSIANGLES, public table 5-97*
PLANE public procedure 5-81
Plane, best fit 5-5
Points in space represented as vectors 4-13
Polypeptides 5-8, 5-54.1, 5-95
Precedence, matrix operators 4-8
Precedence, vector operators 4-8
PRECONNECT public procedure 5-85
PRINT command 2-41
Priorities of ring exit bonds 2-24
PRIORITY light button 2-25
PRIVATE RINGS 2-22
Procedure writing for multimolecule quiz INVOKE 3-93
Procedures 3-1
Procedures, public 5-1
Product, cross 4-6, 4-8
Product, dot 4-6, 4-7
Product, inner 4-6, 4-7
Programming example, INVOKE 3-103
Programming example, molecule 4-25
Programming example, molecules 5-2
Programming example, multimolecule 3-90, 3-103

* indicates an illustration

Page

Programming example, multimolecules 5-48
Programming example, transformers 4-15, 4-25
Programming examples, matrices 4-21
Programming examples, molecules 3-101
Programming examples, vectors 4-21
Programming using molecules 3-1, 1-5
Programming with matrices 4-1
Programming with vectors 4-1
Projection, Newman 3-50*
Properties of molecules 3-46
PROTECT command 2-78
Protecting disk molecules 2-78
Psi torsion angles in polypeptides 5-54.1, 5-97
Public element tables 2-21
Public molecule procedures 5-1
Public procedure CRYST 3-54
Public procedure DISTAB 5-25
Public procedure donation 5-3
Public procedure EDITMODEL 3-65
Public procedure entry, CRYSTQT 5-21
Public procedure entry, MAKEPOLYPEPTIDEQT 5-54.1
Public procedure entry, MOLCONNECT 5-13, 5-22, 5-85
Public procedure entry, SPACEMOLQK 5-103
Public procedure entry, STEREOBOXQT 5-107
Public procedure entry, TABS2MOLQT 5-113
Public procedure, BESTPLANE 5-3
Public procedure, CLEAVEMOL 5-3
Public procedure, CONNECT 5-13, 3-54, 5-22
Public procedure, CONNLIST 5-19
Public procedure, CRYST 5-21
Public procedure, EDITMODEL 5-31, 5-89
Public procedure, FITMOL 5-47
Public procedure, GENMOL 5-51
Public procedure, MAKEPOLYPEPTIDE 5-54.1
Public procedure, MAST 5-55
Public procedure, MBLD 5-57
Public procedure, MFBADDBONDS 5-61
Public procedure, MODIFY 5-63, 5-20
Public procedure, MOLECYCLE 5-67
Public procedure, MOLEDAT 5-69
Public procedure, NEWMAN 5-73
Public procedure, ORMOL 5-77
Public procedure, PLANE 5-81
Public procedure, PRECONNECT 5-85
Public procedure, REFINEMOL 5-89
Public procedure, SETATOMNAMES 5-91
Public procedure, SETMOLNAMES 5-93
Public procedure, SETPEPTIDEBONDS 5-95, 5-9, 5-54.1
Public procedure, SPACEMOL 5-103
Public procedure, STEREOBOX 5-107
Public procedure, TABS2MOL 5-113
Public procedures 5-1, 1-5
Public procedures for computing models 2-28
Public procedures for making molecules 2-16
Public procedures for measuring models 2-57

* indicates an illustration

Page
Public procedures for molecule displays . . . 2-60
Public procedures, MFB . . . B-3
PUBLIC RINGS . . . 2-22
Public table BONDLENGTHS . . . 3-45
Public table of polypeptide torsion angles . . . 5-97
Public table, AMINOACIDS . . . 5-54.2*
Public table, BONDLENGTHS . . . A-5
Public table, DIRECTIONS . . . A-6
Public table, ELEMENTS . . . 5-105, A-2
Public table, PHIPSIANLGES . . . 5-97*
Public tables . . . 1-5, 2-21, 3-43, 5-2, A-1
Public tables, MFB . . . B-1

QCFF energy minimization program . . . 5-89, 2-28, 5-45
Quaternary nitrogen . . . 2-9
Quiz, ADD RING . . . 2-22, 2-28
Quiz, %CONNECTMOL . . . 3-56
Quiz, EDITMODEL . . . 5-32*
Quiz, MAKE MOLECULE . . . 2-3, 1-1, 2-63, 2-75, 3-4, 3-12
3-35, 3-42, 3-62, 5-67, 5-93
Quiz, multimolecule . . . 2-61, 2-79, 3-93
Quiz, Wipke . . . 2-26*

Radius, covalent . . . 3-43
Radius, van der Waals . . . 3-43
Reference frame, multimolecule . . . 3-89
REFINE light button . . . 5-41
REFINE MODEL command . . . 2-28
REFINEMOL public procedure . . . 5-89
Refining molecular models . . . 2-28
Refining molecules . . . 5-41, 5-89
%REFLECTION function . . . 4-14, 4-17
Reflection transformers . . . 4-17*
Resonant bonds . . . 2-10, 2-26, 3-6, 3-51, 3-56
Resonant rings . . . 2-10
RESTORE command . . . 2-14, 2-77
RESTORE light button . . . 2-11
Ring dictionaries . . . 2-21, 2-22, 3-45, A-7
Ring systems . . . 2-21
Rings . . . 2-21, 2-28
Rings, exit bond priorities . . . 2-24
Rings, models . . . 2-21
RINGS, PRIVATE . . . 2-22
RINGS, PUBLIC . . . 2-22
Rings, resonant . . . 2-10
RINGS, SHARABLE . . . 2-22
%RINGSEARCH variable . . . 2-25
%ROTATEONLY function . . . 4-14, 4-17
%ROTATION function . . . 3-90, 4-14, 4-17
Rotation transformers . . . 4-17*

SAVE command . . . 2-14, 2-71, 2-77
Saving disk molecules . . . 2-77
Saving molecules . . . 2-14
SCALE command . . . 2-48, 2-73

* indicates an illustration

Page

Scale factors . . . 4-16
%SCALE function . . . 4-14, 4-16
SCALE light button . . . 2-11, 2-25, 2-27, 2-68, 5-35
Scale transformers . . . 4-16
%SCALEABOUT function . . . 4-14, 4-16
Scaled displays . . . 2-48
%SCALEFAC function . . . 4-14, 4-16
%SCALEV function . . . 4-14, 4-16
Screen coordinates . . . 3-53
SCREEN function . . . 2-20, 3-72, 3-73
Screening molecules . . . 2-18
Searching for substructures . . . 2-17
SELECTING command expression . . . 2-14, 2-44
SELECTING options . . . 3-64
Selective display example . . . 2-45, 5-12, 5-54.5
Selective displays . . . 2-44, 3-4, 5-54.2
Selector columns . . . 3-78, 5-54.2
SET command . . . 2-57, 2-76, 2-82, 3-13
%SETATOMNAME procedure . . . 3-16, 3-20
SETATOMNAMES public procedure . . . 5-91
%SETATOMTYPE procedure . . . 3-16, 3-21
%SETATTRIBUTE example . . . 3-77
%SETATTRIBUTE procedure . . . 3-30, 3-31, 3-91
%SETBONDTYPE procedure . . . 3-24, 3-25
%SETCDERIV procedure . . . 3-78
%SETCONATOMS procedure . . . 3-26, 3-28
%SETCONBONDANGLES procedure . . . 3-67, 3-70
%SETCONBONDLENGTH procedure . . . 3-67, 3-68
%SETCONBONDS procedure . . . 3-26, 3-29, 5-102
%SETCONCHIRAL procedure . . . 3-67, 3-71
%SETCONDIRECTIONS procedure . . . 3-67, 3-69
%SETCONTORSION procedure . . . 3-67, 3-68, 5-102
%SETCONTYPE procedure . . . 3-26, 3-28
%SETCONVALUE procedure . . . 3-26, 3-29, 5-102
%SETFLAG procedure . . . 3-12, 3-32, 3-33
SETMOLNAMES public procedure . . . 5-93
%SETMVEC procedure . . . 3-17, 3-22
SETPEPTIDEBONDS public procedure . . . 5-95, 5-9
SETPEPTIDEBONDS, public procedure . . . 5-54.1
%SETSVEC procedure . . . 3-17, 3-21
Setting model coordinates in multimolecules . . . 3-89
%SETUPATOMNAMES procedure . . . 3-16, 3-19
SHARABLE RINGS . . . 2-22
Sign conventions, torsion angles . . . 4-11*
Single bonds . . . 3-6
Sketch coordinates . . . 3-4, 3-18, 3-21
SKETCH display qualifier . . . 2-41
Sketch of a molecule . . . 2-3, 2-41
Sketching molecules . . . 2-3, 1-1
Solid geometry calculations . . . 4-13
SORT command . . . 2-82
Sorting empirical formulas . . . 2-82, 3-77
Sorting keys, CHEMTEXT . . . 3-77
Space-filling molecule displays . . . 5-103
SPACEMOL public procedure . . . 5-103

* indicates an illustration

Page
SPACEMOLQK, public procedure entry . . . 5-103
Standard bond angles . . . 2-21
Standard bond lengths . . . 2-21
STEREO display qualifier . . . 2-41
Stereo displays of molecules . . . 5-107
STEREOBOX public procedure . . . 5-107
STEREOBOXQT public procedure entry . . . 5-107
Subfield, ATOMS . . . 3-4, 3-2
Subfield, ATTRIBUTES . . . 3-11, 3-2, 3-41, 3-77, 3-84
Subfield, BONDS . . . 3-6, 3-2
Subfield, CONSTRAINTS . . . 3-8, 3-2
Subfield, FLAGS . . . 3-2, 3-12
Subfield, FORMULA . . . 3-12, 3-2, 3-74
Subfield, MOLEC . . . 3-83
Subfield, NAMES . . . 3-12, 3-2, 3-41
Subfield, NOTES . . . 3-12, 3-2, 3-41
Subfield, %SYSCTR . . . 3-84, 3-99
Subfield, %TRANS . . . 3-12, 3-2, 3-41, 3-83, 3-99
Subfields . . . 3-3
Subfields, multimolecule . . . 3-83
Submolecules . . . 3-4
Subscripts in vectors and matrices . . . 4-1
%SUBSEARCH function . . . 3-72, 3-73
Substructure search . . . 2-17, 2-22, 3-72
Substructure search examples . . . 2-19
Subtraction, matrices . . . 4-5, 4-6
Subtraction, vectors . . . 4-5, 4-6
SUPERDELETE command . . . 2-78
SUPERDELETE SAVED command . . . 2-78
%SYSCTR subfield . . . 3-84, 3-99
System-added hydrogens . . . 2-18, 2-41, 2-43, 3-34, 3-62
System bond information . . . 3-45
System element information . . . 3-43
%SYSTEMH function . . . 3-33, 3-34

Tables in molecules . . . 3-3
Tables, public element . . . 2-21
Tablet indications in multimolecule quiz . . . 3-95
TABS2MOL public procedure . . . 5-113
TABS2MOLQT public procedure entry . . . 5-113
%TESTDISKMM function . . . 3-87, 3-92
THREED display qualifier . . . 2-41
Torsion angle constraints . . . 3-68
Torsion angle sign conventions . . . 4-11*
Torsion angles . . . 2-57, 3-50, 4-11
Torsion angles in molecules . . . 2-30
TORSION constraint type . . . 3-10
%TRANS subfield . . . 3-12, 3-2, 3-13, 3-41, 3-83, 3-99
Transformation . . . 1-5
Transformation operator, XFM . . . 4-13, 4-14, 4-19
Transformation programming examples . . . 4-25
Transformer as a matrix . . . 4-13
Transformer building . . . 4-16
Transformer implementation . . . 4-18
Transformer programming example . . . 4-15

* indicates an illustration

Page

Transformer, identity 4-18
Transformer, molecule displays 2-59
Transformers 4-13, 3-12, 3-64
Transformers, combining 4-13
Transformers, display 3-83, 3-84
Transformers, multimolecule 3-99*
Transformers, reflection 4-17*
Transformers, rotation 4-17*
Transformers, scale 4-16
Transformers, translation 4-16
%TRANSLATEV function 4-14, 4-17
%TRANSLATION function 4-14, 4-17
Translation transformers 4-16
%TRANSPOSE function 4-9, 4-10
Transposition of a matrix 4-9
Triple bonds 3-6
%TRIPOD function 4-10, 4-12
TURN command 2-49, 2-73
TURN example 2-52*
TURN light button 2-11, 2-25, 5-34
TWIST command 2-53, 2-73
TWIST light button 2-70, 5-41
Twisting molecules 3-63
%TWISTMOL procedure 3-55, 3-63
Type, atom 3-4
%TYPEMOL example 3-48
%TYPEMOL procedure 3-46, 3-47
Types of atoms 3-20
Types of constraints 3-8
Types, atom 3-43
Types, bond 3-25
Typing molecule information 3-46

UNBIND command 2-76
Unit vectors 4-9
%UNITVEC function 4-9, 4-10
UNMARK command 2-45, 3-78
%UNMARKATOMS procedure 3-79, 3-80
%UNMARKBONDS procedure 3-79, 3-81

Valences 3-51
Van der Waals radius 3-43, 5-104
Vector addition 4-5, 4-6
Vector arithmetic 4-5
Vector arithmetic operators 4-5
VECTOR data type 3-18, 4-1
Vector division 4-7
Vector functions 4-9
VECTOR light button 2-71
Vector magnitude 4-9
Vector multiplication 4-5, 4-6
Vector negation 4-5, 4-6
Vector operator precedence 4-8
Vector programming 4-1
Vector programming examples 4-21

* indicates an illustration

Page

Vector subtraction 4-5, 4-6
Vectors 1-5
Vectors and matrices, array constants 4-1
Vectors and matrices, subscripts 4-1
Vectors as directions 4-13
Vectors as points in space 4-13
Vectors, perpendicular 4-9
Vectors, unit . . . . ;4-9
%VERIFYMOLCONNS example 3-52
%VERIFYMOLCONNS procedure 3-47, 3-52

Weight, molecular 3-48
WIPKE column heading 3-4, 3-6, 3-41
WIPKE energy minimization program 2-21, 2-25, 3-4, 3-6
Wipke quiz 2-26*
%WIPKEFLAG variable 2-25

XCOORDINATE command expression 2-55
XFM operator 4-13, 4-19

YCOORDINATE command expression 2-55

ZCOORDINATE command expression 2-55

%ADDATOM example 3-23
%ADDATOM procedure 3-16, 3-18
%ADDBOND example 3-23
%ADDBOND procedure 3-23, 3-24
%ADDCONSTRAINT procedure 3-26, 3-27
%ADDNEWATOM example 3-63
%ADDNEWATOM function 3-16, 3-18
%ADIHEDRAL function 3-47, 3-50
%ADJUSTMOL procedure 3-38, 3-18, 3-21, 3-22, 3-32
3-39, 3-65, 3-77, 3-88, 3-89
%ADJUSTMULTIMOL procedure 3-86, 3-88, 3-89, 3-91, 3-95
%ANGLE3 function 4-10, 4-11
%ANGLE4 function 4-10, 4-11
%ATOMNAME2INDEX function 3-16, 3-20
%ATOMNAMES2INDICES function 3-16, 3-20
%ATOMNEIGHBORS function 3-33, 3-34
%ATOMSDIST function 3-47, 3-49
%AUTOBAT procedure 5-54.6, 5-89
%BDIHEDRAL function 3-47, 3-50
%BONDBETWEEN function 3-32, 3-33
%BONDEDCOUNT function 3-47, 3-51
%CHEMCTNS function 3-72, 3-73
%COMPOSITION function 3-46, 3-47
%COMPUTEATOMSANGLE function 3-47, 3-49
%CONNECTMOL example 3-57*
%CONNECTMOL procedure 3-54, 3-55, 5-17, 5-22, 5-70, 5-85
%CONNECTMOL quiz 3-56
%CONNT2A procedure 3-47, 3-51
%CONVCHEMTEXT function 3-75, 3-76
%CONVCHEMTEXTQT function 3-75, 3-76
%CONVERT2COREMM procedure 3-87, 3-91

* indicates an illustration

Page

%CONVERT2COREMOL procedure . . . 3-39
%CONVERT2DISKMM procedure . . . 3-86, 3-91
%CONVERT2DISKMOL procedure . . . 3-39, 3-42
%CRYSTXFM function . . . 4-14, 4-18
%DBOX procedure . . . 3-53
%DELETEATOM procedure . . . 3-16, 3-18
%DELETEATOMS procedure . . . 3-16, 3-18
%DELETEATTRIBUTE procedure . . . 3-30, 3-31, 3-91
%DELETEBOND procedure . . . 3-24, 3-25
%DELETECONSTRAINT procedure . . . 3-26, 3-27
%DET function . . . 4-10, 4-11
%DTEXT procedure . . . 3-53
%FINDBONDANGLE function . . . 3-47, 3-49
%FINDBONDLENGTH function . . . 3-44, 3-45
%FINDELEMENT procedure . . . 3-43, 3-44
%GETATOM1 function . . . 3-24, 3-25
%GETATOM2 function . . . 3-24, 3-25
%GETATOMNAME function . . . 3-16, 3-19
%GETATOMTYPE function . . . 3-16, 3-20
%GETATTRIBUTE function . . . 3-30, 3-31, 3-91
%GETBONDA function . . . 3-16, 3-21, 3-34
%GETBONDTYPE function . . . 3-24, 3-25
%GETCHEMTEXTKEY function . . . 3-75, 3-77
%GETCONATOMS function . . . 3-26, 3-28
%GETCONBONDLENGTH function . . . 3-66, 3-67
%GETCONBONDS function . . . 3-26, 3-28
%GETCONCHIRAL function . . . 3-67, 3-70
%GETCONDIRSANGLES function . . . 3-67, 3-69
%GETCONTORSION function . . . 3-67, 3-68
%GETCONTYPE function . . . 3-26, 3-28
%GETCONVALUE function . . . 3-26, 3-29
%GETFLAG function . . . 3-12, 3-32, 3-33
%GETMVEC function . . . 3-17, 3-22
%GETNEWMVEC example . . . 3-63
%GETNEWMVEC function . . . 3-55, 3-62
%GETRINGMOL function . . . 3-44, 3-45
%GETSVEC function . . . 3-16, 3-21
%HBALANCE procedure . . . 3-62, 3-35, 3-55
%IDENTITYMATRIX function . . . 4-10, 4-12
%IDENTITYXFM constant . . . 4-14, 4-18
%INVERSE function . . . 4-9, 4-10
%INVXFM function . . . 3-100, 4-14, 4-18
%LASTATOM function . . . 3-16, 3-19
%LASTBOND function . . . 3-24, 3-25
%LASTCONSTRAINT function . . . 3-26, 3-28
%MAGSQD function . . . 4-9, 4-10
%MAKEMOL procedure . . . 3-38, 3-39
%MAKEMOLSELECT procedure . . . 3-39, 3-40
%MAKEMULTIMOL procedure . . . 3-85, 3-86
%MARKATOMS example . . . 5-10
%MARKATOMS procedure . . . 3-40, 3-79, 3-80
%MARKBONDS procedure . . . 3-40, 3-79, 3-81
%MATDIMEN procedure . . . 4-10, 4-12
%M%DSK flag . . . 3-12
%MERGEMOL procedure . . . 3-55, 3-64

Page

%M%FRAG flag 3-12
%MMADDMOL procedure 3-86, 3-88
%MMDELETEMOL procedure 3-86
%M%MDL flag 3-12, 3-40
%MMGETMOL function 3-86, 3-88
%MMGETMVEC function 3-89, 3-90, 3-96, 3-98
%MMGETMXFM function 3-94, 3-97, 3-100
%MMLASTMOL function 3-86, 3-88
%MMMANIPMOL procedure 3-86, 3-90, 3-98
%MMQMULTIMOL function 3-94, 3-95
%MMQREFRESH procedure 3-94, 3-95
%MMSETMVEC function 3-89
%MMSETMXFM procedure 3-94, 3-97
%MOLBONDLENGTH function 3-47, 3-49
%MOLCOORDS procedure 3-47, 3-53
%MOLCOPY function 3-39, 3-40, 3-89
%MOLWEIGHT function 3-47, 3-48
%M%SKT flag 3-12, 3-40
%MULTIMOLCOPY function 3-85, 3-86
%M%UNBD flag 3-12
%OTHERATOM function 3-33, 3-34
%PERPTO function 4-9, 4-10
%PERSPECTIVE 5-104
%PERSPECTIVE procedure 2-58, 2-73
%PERSPECTIVEQK procedure 2-60, 1-2
%REFLECTION function 4-14, 4-17
%RINGSEARCH variable 2-25
%ROTATEONLY function 4-14, 4-17
%ROTATION function 3-90, 4-14, 4-17
%SCALE function 4-14, 4-16
%SCALEABOUT function 4-14, 4-16
%SCALEFAC function 4-14, 4-16
%SCALEV function 4-14, 4-16
%SETATOMNAME procedure 3-16, 3-20
%SETATOMTYPE procedure 3-16, 3-21
%SETATTRIBUTE example 3-77
%SETATTRIBUTE procedure 3-30, 3-31, 3-91
%SETBONDTYPE procedure 3-24, 3-25
%SETCDERIV procedure 3-78
%SETCONATOMS procedure 3-26, 3-28
%SETCONBONDANGLES procedure 3-67, 3-70
%SETCONBONDLENGTH procedure 3-67, 3-68
%SETCONBONDS procedure 3-26, 3-29, 5-102
%SETCONCHIRAL procedure 3-67, 3-71
%SETCONDIRECTIONS procedure 3-67, 3-69
%SETCONTORSION procedure 3-67, 3-68, 5-102
%SETCONTYPE procedure 3-26, 3-28
%SETCONVALUE procedure 3-26, 3-29, 5-102
%SETFLAG procedure 3-12, 3-32, 3-33
%SETMVEC procedure 3-17, 3-22
%SETSVEC procedure 3-17, 3-21
%SETUPATOMNAMES procedure 3-16, 3-19
%SUBSEARCH function 3-72, 3-73
%SYSCTR subfield 3-84, 3-99
%SYSTEMH function 3-33, 3-34

Page

%TESTDISKMM function 3-87, 3-92
%TRANS subfield 3-12 3-2, 3-13, 3-41, 3-83, 3-99
%TRANSLATEV function 4-14, 4-17
%TRANSLATION function 4-14, 4-17
%TRANSPOSE function 4-9, 4-10
%TRIPOD function 4-10, 4-12
%TWISTMOL procedure 3-55, 3-63
%TYPEMOL example 3-48
%TYPEMOL procedure 3-46, 3-47
%UNITVEC function 4-9, 4-10
%UNMARKATOMS procedure 3-79, 3-80
%UNMARKBONDS procedure 3-79, 3-81
%VERIFYMOLCONNS example 3-52
%VERIFYMOLCONNS procedure 3-47, 3-52
%WIPKEFLAG variable 2-25

/*** Update to the PROPHET Molecules Manual, February 1981 ***/

TABLE OF CONTENTS

Chapter II Updates: Molecule Commands

1. NOHBALANCE qualifier in MAKE MOLECULE
2. Comparison of ignored constraint values to actual computed model values in COMPUTE MODEL command
3. DISPLAY ... FITTED/NOFITTED and DISPLAY ... SCALEFAC/NOSCALEFAC
4. Change in MARK (or UNMARK) ATOMS and BONDS quizzes

Chapter III Updates: Molecule Functions and Procedures

1. %DELETEBONDS Procedure
2. %COMPUTECHIRALITY procedure
3. %SETALLCONSTRAINTS Procedure
4. %ADDMOL Procedure
5. %GETATOMINDEX Function
6. %GETBONDINDEX Function
7. %BOXATOM Procedure

Chapter V Updates: Public Molecule Procedures

1. NEWMAN
2. STEREOBOX
3. LOGP

Chapter II: Molecule Commands

1. NOHBALANCE qualifier in MAKE MOLECULE

The MAKE MOLECULE command has been expanded by the addition of a qualifier called NOHBALANCE. If you do not wish to have the MAKE MOLECULE quiz automatically add hydrogens to the carbon and nitrogen atoms in a molecule you are making, you can use a command like:

```
MAKE MOLECULE M NOHBALANCE
```

where M is the name of the molecule in this example. You will then be responsible for adding hydrogens as needed yourself within the quiz. This is useful when you have a molecule where you wish to represent one or more of the carbon or nitrogen atoms as not obeying the usual valence rules. When you EXIT from the quiz, PROPHET still complains about any bonding patterns that it does not recognize (it would, for example, complain about a carbon atom with only three single bonds), but it does not force you to make changes in the bonding patterns.

It is also possible to use a command like:

```
MAKE MOLECULE M2 FROM M1 NOHBALANCE
```

This command would put you in the MAKE MOLECULE quiz with molecule M1 shown. Molecule M2, which you use the quiz to create from M1, will contain all of the system added hydrogens that molecule M1 had, but no newly added carbon or nitrogen atoms will be hydrogen balanced.

Another example of the use of this qualifier is in a command like:

```
EDIT M NOHBALANCE
```

After editing molecule M, the only system added hydrogens that will be in the molecule are those that were already there before you typed the EDIT command.

You can use the NOQUIZ and NOHBALANCE qualifiers in the same command, but NOQUIZ must follow NOHBALANCE. For example, the command:

```
MAKE MOLECULE M2 FROM M1 NOHBALANCE NOQUIZ
```

would make a new molecule M2 that is an exact copy of molecule M1 without adding any hydrogens beyond those that were already present in M1 and without going to the MAKE MOLECULE quiz.

2. Comparison of ignored constraint values to actual computed model values in COMPUTE MODEL command

The COMPUTE MODEL command ignores most constraints affecting atoms and bonds in ring systems, as described on pages 2-28 to 2-39 of this manual. If any such constraints are ignored by the model builder, you are informed of this fact and asked if you wish to see the actual computed values for the parameters you tried to constrain. If you answer YES to this question, you now receive a display of a table showing the constraint values that you set as well as the actual values in the computed model.

3. DISPLAY ... FITTED/NOFITTED
DISPLAY ... SCALEFAC/NOSCALEFAC

The FITTED qualifier is used to make a molecule display fill as much of the available space on the screen as possible. For example, the command:

```
EAD M FITTED<GO>
```

causes the display of the sketch of the molecule M to be expanded or shrunk so that it occupies the entire screen, but does not extend beyond the screen's edges. Similarly, the command

```
DISPLAY M THREED FITTED<GO>
```

causes the three-dimensional model of M to be displayed within whatever space remains on the screen, along with the scale factor that was involved in getting the display to fit this space. The system standard default for this qualifier is NOFITTED, which displays the molecule without affecting its scale factor.

The NOSCALEFAC qualifier is used when you wish to display the three dimensional model of a molecule that has been scaled without showing what the scale factor is. For example, the command:

```
EAD M THREED COMPLETE FITTED NOSCALEFAC<GO>
```

results in a display of M's model (complete with hydrogens) that fills the screen but does not show the scale factor that was required to make the display this size. The system standard default for this qualifier is SCALEFAC, which causes the scale factor to be displayed along with the model of any molecule that has been scaled, as in the past.

The DEFAULT MOLECULES command described on page 2-43 of the PROPHET Molecules manual may be used to change the default settings for these display qualifiers as well as for any of the other molecule display qualifiers. For example, the command:

```
DEFAULT MOLECULES TO THREED FITTED NOSCALEFAC<GO>
```

causes all subsequent molecule displays to show three dimensional models fitted to the space on the screen, without showing the scale factor, unless you specify otherwise. These two pairs of qualifiers have also been added to the DEFAULT MOLECULES quiz illustrated on page 2-44 of this manual.

4. Change in MARK (or UNMARK) ATOMS and BONDS quizzes

A minor change has been made in the quizzes that follow the MARK and UNMARK commands described on pages 2-44 to 2-47 of the PROPHET Molecules manual. (These are the commands that you use to "mark" a group of atoms or bonds in a molecule so that group can be displayed by itself - using the DISPLAY ... SELECTING command - or a new molecule can be made that includes that group - using the MAKE MOLECULE ... FROM ... SELECTING command.) In the past, if you typed the MARK or UNMARK command at a time when the molecule in question was not already displayed on the screen, the molecule would be displayed using the standard display qualifiers (i.e., SKETCH NOCOMPLETE NODREIDING NOATOMINDEX NOBONDINDEX TYPES NOFITTED). In the new version, you will get the same standard display, unless you have used the DEFAULT MOLECULES command (page 2-43 of PROPHET Molecules) to specify your own set of molecule display qualifiers, in which case MARK and UNMARK will display the molecule with the default qualifiers you have specified. If, for example, you type the commands:

```
DEFAULT MOLECULES TO FITTED THREED COMPLETE<GO>
MARK ATOMS OF M FOR 'GROUP 1'<GO>
```

PROPHET will display the three-dimensional model of molecule M complete with added hydrogens and fitted to the size of the screen for you to use in marking the atoms that you consider to belong to 'GROUP 1'.

Chapter III: Molecule Functions and Procedures

1. %DELETEBONDS procedure

The procedure %DELETEBONDS is similar to the procedure %DELETEATOMS described on page 3-18 of this manual. It is called by a statement like:

```
CALL %DELETEBONDS(M,BONDLIST)
```

where M is the MOLECULE in question, and BONDLIST is a FIXED ARRAY of the indices of the bonds to be deleted. Unlike %DELETEATOMS, which deletes any bonds connected to the atoms that are being deleted, %DELETEBONDS never deletes any atoms. Both %DELETEATOMS and %DELETEBONDS delete any constraints that refer to any atoms or bonds that are deleted. Note that it is much more efficient to call %DELETEBONDS once when you are deleting several bonds from a molecule than it is to make multiple calls to %DELETEBOND.

2. %COMPUTECHIRALITY procedure

%COMPUTECHIRALITY is a procedure that returns the chirality, CC or CW, about an atom of a molecule (see page 2-37 of this manual for a description of what CC and CW indicate in PROPHET). It is called with a statement like the following:

```
CHIRAL = %COMPUTECHIRALITY(M,AIND,BONDA)
```

The first argument, M, is a MOLECULE, AIND is the FIXED index of the atom about which the chirality is to be determined, and BONDA is a FIXED ARRAY of bond indices. The bond index array must have three entries which are the indices of three of the four bonds joining the atom whose index is AIND to its neighboring atoms. The bond array is equivalent to that found in the constraints table for a chirality constraint. The chirality returned as the TEXT values of the function (either 'CC' or 'CW') is that looking toward the atom of the second argument from the fourth neighboring atom (the one not at the end of one of the three bonds in the bond array).

If the chirality cannot be determined, such as when the molecule does not have a model or atom AIND does not have exactly four bonds attached to it, then the procedure returns an empty value.

3. $SETALLCONSTRAINTS procedure

$SETALLCONSTRAINTS is a procedure that enables you to set up a CONSTRAINTS table for a molecule that already has a model so that you can edit the molecule and then compute a new model without changing the model for the unmodified sections of the molecule. It should prove useful in cases where the model coordinates are derived from a source other than COMPUTE MODEL (e.g., crystallographic coordinates). It sets all of the 'DIRECTIONS', 'BONDLENGTH', and 'TORSION' constraints for a molecule. It is called with a statement like the following:

```
CALL $SETALLCONSTRAINTS(M)
```

The argument M is of type MOLECULE and is the molecule for which the constraints are to be set.

This procedure does the following:

i. For each atom: If there is no 'BOND ANGLES' or 'DIRECTIONS' constraint, and the connected atoms all have model coordinates, a 'DIRECTIONS' constraint is set.

ii. For each bond: If there is no 'BONDLENGTH' constraint and the atoms at the ends of the bond have model coordinates, a 'BONDLENGTH' constraint is set. If there is no 'TORSION' constraint, and the connected atoms have model coordinates, a 'TORSION' constraint is set.

Also, if the estimated number of constraints is greater than 150 and the molecule is core based, it is converted to disk based. Any 'CHIRALITY' constraints that exist for atoms that also have either 'BOND ANGLES' or 'DIRECTIONS' constraints are deleted.

NOTE: Although this procedure computes all constraints for a molecule, at the present time, COMPUTE MODEL ignores most constraints in rings.

4. %ADDMOL Procedure

%ADDMOL is a procedure that adds a fragment molecule to a "parent" molecule. It is called with a statement like the following:

```
CALL %ADDMOL(PMOL,FMOL,PATOMS,FATOMS,DIHED,BTYPE,NEWF2)
```

The arguments have the following types and meanings:

Argument	Type	Meaning
PMOL	MOLECULE	"Parent" molecule to be added to
FMOL	MOLECULE	Fragment molecule to add
PATOMS	FIXED ARRAY	Indices of three atoms in parent molecule that determine where the fragment will be added
FATOMS	FIXED ARRAY	Indices of three atoms in fragment molecule that determine how fragment is added
DIHED	FLOAT	Dihedral angle to be formed by atoms PATOMS[3] - PATOMS[1] - FATOMS[1] - FATOMS[3]
BTYPE	TEXT	Type of bond joining parent and fragment
NEWF2	FIXED	Set by %ADDMOL to new index in parent molecule of atom whose index was F2ATOMS[2] in fragment

For convenience in the following description, we will refer to the atom whose index is PATOMS[1] as atom P1. Likewise, we will call PATOMS[2] atom P2, PATOMS[3] atom P3, FATOMS[1] atom F1, FATOMS[2] atom F2 and FATOMS[3] atom F3.

The key to the function of %ADDMOL is that the parent molecule must have an "exit bond" that determines where the fragment is added according to the fragment's own "exit bond," which is superimposed on the parent's exit bond. If the three specified atoms in the parent molecule are arranged as follows:

```
                  --P3
rest of parent        \
                    --P1----P2
```

and the fragment's three key atoms are similarly arranged:

```
             F2----F1--
                     \      rest of fragment
                      F3--
```

then the modified parent molecule that results from calling

%ADDMOL will have the following structure within it:

```
      --P3
          \
       --P1----F1--
                 \
                  F3--
```

Atom F2 will also be present, superimposed on atom P1, and atom P2 will be located somewhere along the bond connecting atoms P1 and F1. You should add the index of atom P2 and the new index of atom F2 (returned in tne argument NEWF2) to a FIXED ARRAY of atom indices that you use in a call to %DELETEATOMS after you are done with all your calls to %ADDMOL for adding fragments to a particular parent molecule.

%ADDMOL joins both the sketch and tne model (if both tne parent and the fragment have models) of the fragment to the parent molecule. It functions by superimposing atom F2 on atom P1 and then rotating the fragment so tnat the bond between atoms F1 and F2 is superimposed on tne bond between atoms P1 and P2. A new bond of type BTYPE is added to join atom P1 to atom F1, and the final adjustment to the model coordinates involves twisting the fragment about tne new joining bond so that the dihedral angle formed by atoms P3 - P1 - F1 - F3 is the angle specified by DIHED. Note tnat this series of steps results in giving the new bond joining atom P1 to atom F1 a length equal to tne length of tne bond that originally joined atom F1 to atom F2 in the fragment.

%ADDMOL uses a system entry to tne procedure %MERGEMOL to actually add the fragment to the parent molecule. Hence any special columns (such as atom name, selecting or labeling columns) in tne fragment's ATOMS or BONDS table are added to the parent molecule's ATOMS and BONDS tables. Similarly, all entries in tne fragment's CONSTRAINTS table are added to tne parent molecule's CONSTRAINTS table. Note that any constraints that refer to tne fragment's exit bond (the bond joining atom F1 to atom F2) will be deleted when you call %DELETEATOMS after you are done calling %ADDMOL.

Unlike %MERGEMOL, %ADDMOL does not attempt to preallocate the ATOMS, BONDS and CONSTRAINTS tables of tne parent molecule to accomodate the enlarged size of tne modified molecule including the added fragment, nor does it call %ADJUSTMOL after the fragment has been added. Therefore, for efficiency considerations, it is wise for you to preallocate the sizes of the parent molecule's tables before your first call to %ADDMOL, and you must include a call to %ADJUSTMOL after your %DELETEATOMS call following your last call to %ADDMOL.

The structure of a typical procedure that repetitively calls %ADDMOL to add fragments to a growing parent molecule (such as a polypeptide) is illustrated in tne following incomplete sample procedure:

```
          ADDFRAGS:  PROCEDURE(P,FRAGS,DIHED);

DECLARE P       MOLECULE,          /* Parent molecule */
        FRAGS   MOLECULE ARRAY,    /* Fragments to be added */
         DDIHED   FLOAT;            /* Dihedral angle at each
                                      joining bond */
DECLARE PA      FIXED ARRAY,       /* Atom indices used in
                                      %ADDMOL call */
        FA      FIXED ARRAY,       /* Fragment atom indices
                                      used in
                                      %ADDMOL call */
        NDEL    FIXED,             /* Number of atoms to delete */
        DLIST   FIXED ARRAY,       /* List of atoms to delete */
        TATOMS  FIXED,             /* Total atoms in P plus all
                                      fragments */
        TBONDS  FIXED,             /* Total bonds in P plus all
                                      fragments */
        I       FIXED;             /* DO index */

 NDEL = 0;        /* Initialize NDEL */

 /* Preallocate ATOMS and BONDS tables of P for efficiency */

 TATOMS = %LASTATOM(P);
 TBONDS = %LASTBOND(P);

 DO I = 1 TO TOP(FRAGS);
     TATOMS = TATOMS + %LASTATOM(FRAGS[I]);
     TBONDS = TBONDS + %LASTBOND(FRAGS[I]);
 END;     /* DO I to add numbers of atoms and bonds in each
             fragment to total */

 ALLOCATE P.ATOMS(TATOMS,%LASTCOL(P.ATOMS));
 ALLOCATE P.BONDS(TBONDS,%LASTCOL(P.BONDS));

 DO I = 1 TO TOP(FRAGS); /* Loop to add each fragment */

     PA[1] =   /* Do whatever is appropriate to get  */
     PA[2] =   /* joining atom indices for parent    */
     PA[3] =

     FA[1] =   /* Do whatever is appropriate to get */
     FA[2] =   /* joining atom indices for fragment */
     FA[3] =

     CALL %ADDMOL(P,FRAGS[I],PA,FA,DIHED,DLIST[NDEL+2]);

     DLIST[NDEL+1] = PA[2];      /* Update deletion list */
     NDEL = NDEL + 2;

 END;    /* DO I for each fragment to be added */

 /* Delete excess atoms and adjust molecule */
```

```
CALL %DELETEATOMS(P,DLIST);
CALL %ADJUSTMOL(P);

END;    /* of sample procedure ADDFRAGS */
```

5. %GETATOMINDEX function

A new function, %GETATOMINDEX, is available that returns the index of the atom the user points to in a displayed molecule. It is called with a statement like:

```
AIND = %GETATOMINDEX(M,PROMPT)
```

where M is the MOLECULE, and PROMPT is a TEXT that %GETATOMINDEX prompts the user with. For example, the statement:

```
AIND = %GETATOMINDEX(M,'POINT TO AN ATOM')
```

will first display molecule M on the screen using the current default display qualifiers (unless the molecule is already displayed) and then type

```
POINT TO AN ATOM
```

It will then wait until the user has pointed to an atom in molecule M, using the pen and data tablet, giving appropriate error messages if the user points to anything other than an atom in the display of M. When the user has pointed to an atom, the index of the atom pointed to is typed and the FIXED value of that index is returned (and assigned to the variable AIND in this example); if the user pushes <GO>, an EMPTY value is returned.

6. %GETBONDINDEX function

%GETBONDINDEX is similar in all respects to %GETATOMINDEX, except that it returns the index of the bond in the argument molecule that the user points to.

7. $BOXATOM procedure

The new system procedure $BOXATOM draws a box around an atom in a displayed molecule. It is called with a statem ent like:

```
CALL $BOXATOM(M,AIND)
```

where M is the displayed molecule and AIND is tne FIXED index of the atom to draw the box around. If you want to draw a box around an atom that the user has pointed to, you could use a statement like:

```
CALL $BOXATOM(M,$GETATOMINDEX(M,'POINT TO AN ATOM'))
```

Chapter V: Public Molecule Procedures

1. NEWMAN

The public procedure NEWMAN now has another entry point, NEWM ANANGLE. The calling sequence is:

```
CALL NEWMANANGLE(mol-name, atom0, atom1, atom2, atom3)
                <molecule><fixed><fixed><fixed><fixed>
```

NEWMANANGLE will produce the Newman projection looking from atom1 toward atom2 similar to NEWMAN. However, NEWMANANGLE will also rotate the projection until atom0 is at the top and display the angle between atom0 and atom3. Atom0 must be a neighbor atom of atom1 and atom3 must be a neighbor atom of atom2. All four atoms must be specified by their numeric index. Like NEWMAN, NEWMANANGLE only works on display terminals and the molecule must have a 3D model.

2. STEREOBOX

STEREOBOX works for multimolecules as well as molecules. The way it is called is the same as for molecules (see page 5-107 of this manual). The only difference in using STEREOBOX on a molecule and on a multimolecule is that, in the multimolecule case, you cannot selectively display atoms.

3. LOGP

Questions on the Public Procedure LOGP should be directed to Howard Johnson or Kay Leibrand at SRI or Michael Cory at Burroughs-Wellcome.

LOGP is a procedure to compute partition coeffients for molecules based on the method described by Corwin Hansch and Albert Leo in "Substituent Constants for Correlation Analysis in Chemistry and Biology" (Wiley, 1979). There are some differences between the method in the reference and the method used by the procedure and these will be mentioned in the following description.

The procedure operates on tables of molecules using the entries

LOGP (interactive)

or

LOGPQT(T,TOUTID,CNM,CNMN,FRN,LRN) (quiet).

First time users should try the interactive entry first.

The arguments for the quiet entry are:

T	Table	The input table of molecules it must be dearchived and accessed.
TOUTID	Text	Name for the output table Enclose it in quotes to make it a text.
CNM	Fixed	column number for molecules
CNMN	Fixed	Column number for molecule names
FRN	Fixed	Number of first row to be processed
LRN	Fixed	Number of last row to be processed.

It requires the public table LOGPFRAGS which contains the necessary fragments and corresponding constants.

Notes on using the log P calculating program. (Cory's Chemical Caveats)

1. Tests on sets of varied compounds gives good results. (A standard deviation of .3 log can be expected.)

2. Certain areas must be checked carefully. The program warns the user in all cases where it can not find atoms. If this type of "partial" occurs the only recourse is to estimate the value based upon similar groups in the data tables.

All fragments must be drawn as they are specified in the fragment table (LOGPFRAGS). In particular PROPHET will allow many representations of groups such as nitro . Only one representation will be matched and the others will give errors. The best way to check this is to look at the fragment in the fragment table using a display command and draw your molecules to match.

Empty cells in the fragment table are also flagged by the program. Again the only recourse is to estimate the value using the rules.

Partials will occur because of unimplemented rules. Usually this is because of a Halogen Polar interaction. these can be estimated from the very incomplete data in Hansch's book.

Compounds with aromatic structures (as defined by Hansch for these purposes) can also cause problesm. This can best be handled by looking at model compounds in the literature and attempting to adjust the LOG P estimate to fit the ring system in question.

There are also special cases (i.e., morpholine, piperidine,epoxide) in which Hansch has defined special factors. These special factors have not been accounted for to date. In general it is a significant factor (i.e., a difference of .3 log or more) for these compounds and should be checked.

General Procedure Flow

1) Get table of molecules, column numbers for molecules and molecule names, and first and last row numbers of the section to be done.

2) Get display option. If the procedure is run interactively, it is possible to have the molecules displayed and the atoms boxed as they are found by the substructure search. This increases the run time by about 15%.

3) Make output table, if required.

4) Get fragment table and section definitions. (see separate description of the fragment table.)

5) Compute LOGP for each molecule and store result in output table. (See description below.)

Note: Because of the fixed time required for the "set-up" decscribed above, it is more efficient to use the procedure on longer tables of molecules.

For each molecule:

a) Gets a copy of the molecule from the table.

b) Converts appropriate S and D bonds to R bonds in the molecule. It uses column WIPKE of the BONDS table if available; otherwise it uses a set of rules described below. It does not change the bond types in the molecule in the table, but does change the WIPKE column of the molecule in the table for future use.

To determine which S and D bonds should be changed to R bonds, each fused ring system in the molecule is examined. If the number of atoms in the system is equal to twice the number of double bonds in the system, then all bonds in the system are considered to be aromatic. If a fused system fails the atom-double bond test above, then each ring in the system is examined separately. If each atom in a ring has a double bond in the ring or a leaving R or DR bond, then the ring is considered to be aromatic. This rule is applied to each system iteratively until there are no more bond types to be changed.

After the above tests, the system is checked for 5-membered rings. A 5-membered ring is considered to be aromatic if it has an N,S or O atom with 2 single bonds in the ring and if these 2 single bonds are connected to D or R bonds in the ring.

Finally, any single bond directly connecting aromatic systems is considered aromatic.

Note that this definition of aromaticity is somewhat different from that of the reference.

c) Sets up a column in the atoms table to "check-off" atoms as they are found and counts the atom types and numbers of atoms of each type.

d) Makes a MOLSCREEN of the molecule.

e) If there are R bonds in the molecule, then searches (using the PROPHET substructure search algorithm) each section of aromatic fragments in the fragment table corresponding to atom types actually in the molecule. Keeps track of the fragments found and where (in atom index arrays), checks off found atoms and keeps atom counts current.

6) Searches each section of non-aromatic fragments (as in 5 above).

7) Identifies all carbon and hydrogen atoms which were not found as parts of fragments as aliphatic.

8) Classifies the bonds to be used in determining fragment constants and special factors.

All fragment bonds are classified as FRAG (internal to fragment), NP (non-polar leaving bond) or PLR (polar leaving bond) with PLR taking prededence if more than one condition applies. (The determination of polar or non-polar is taken from column 'POLAR' of the fragment table.)

All ring bonds are then classified as AROMRNG (an R bond in a ring) or ALRNG (a non-R bond in a ring) or, if they are also fragment bonds, FRAG, PLR or NP with AROMRNG or ALRNG concatenated.

All hydrogen bonds are then classified as H. This classification overrides any previous classification.

Any remaining bonds are classified as NP.

Some of these classifications are changed by subsequent procedures.

9) Computes, types and adds factors to appropriate totals for:

a) Geminal and vicinal halogenations. Examines carbons which are not internal to found fragments for halogen connections. Note that these include aliphatic but not aromatic ring carbons. Geminal halogenation is the occurrence of 2,3, or 4 halogens on one of these carbons. Halogens are vicinally related if they are attached to adjacent carbon atoms which are connected by single bonds. A factor is applied for each halogen in the case of geminal halogenation and for N-1 of N vicinally related halogens. The bond classification of bonds to such halogens is set to HAL so that they will not be considered in chain branching.

b) Chain bonds, chain branches and group branches. Any bond which has been classified as PLR, NP or HAL by the bond classification described above is considered to be a chain bond unless it is isolated, i.e., connected to no other chain bonds through a non-ring atom) on a ring. A factor of -0.12 is applied for each of N-C chain bonds where C is the number of separate chains (i.e., "clumps" of connected chain bonds).

To determine branching, the connections to each carbon which has not been found as part of a fragment are examined. The procedure determines NPB (the number of PLR bonds) and NNPB (the number of NP bonds plus the number of aliphatic ring bonds to a maximun of 2). If the total of these two types of bonds is less than 3, then there is no branching.

If NNPB>2 then the number of chain branches at the carbon in question is NNPB-2; otherwise there is no chain branch at that carbon.

The number of group branches at the carbon in question is equal to the number of PLR bonds at that carbon unless NNPB<2. (i.e., unless there are not enough NP or aliphatic ring bonds to make a chain through the carbon). In this case the number of group branches is NPB-(2-NNPB) (i.e., enough PLR bonds are taken to make a chain through the carbon and the rest are taken as group branches)

c) Nitrogen special branch factors for non-polar chains. The branch types handled are tertiary, quaternary with no hydrogens, quaternary with 1 hydrogen, and quaternary with 2 or 3 hydrogens. Chain bonds are counted from the nitrogen up to but not including the first polar bond. Geomeric and electronic effects are calculated according to the tables on page 39 of the reference and there are no adjustments for special cases. The factors do not include the basic chain bond factor of -0.12 which is added in when chain bonds are counted.

d) Aliphatic ring bonds which are not internal to found fragments.

e) Ring clusters--a fused ring system is a ring cluster if it has at least 2 rings which have no polar bonds and if these rings have at least 2 adjacent aliphatic bonds in common.

f) Double and triple carbon-carbon bonds of types normal or directly conjugated to one or two aromatic systems. The bond classification for a carbon-carbon double bond is changed to CCDB if it is not conjugated to an aromatic system and CCDBPHEN if it is.

The factors are different from Leo's because Leo considers the structure saturated and uses the bond factor to correct for this whereas the procedure uses an exact hydrogen count. The methods are equivalent.

g) Checks the attributes of the molecule for the existence of intra-molecular H-bonds and, if any, adds the appropriate factor to the total partition coefficient. Specifically, it checks the 'H-BOND' attribute of the molecule to see if it is either 'nitrogen' or 'oxygen' and applies the factor 0.60 or 1.0. Note that it is assumed that there is only one intra-molecular H-bond and that it is one of the above two types.

10) Assigns constants to each found fragment, taking constants from the appropriate column of the fragment table or computing constants for special cases.

If any of the constants needed for a particular calculation is not in the table, then the fragment is assigned an empty value and the total partition coefficent will be labeled as partial.

a) For aromatic fragments, the constant in column 3 of the fragment table is used with the exception of nitrogens and ring fusion carbons. for a ring fusion carbon, if any atom connected to it is not a carbon then the constant in column 4 (i.e., CSTAR) is used; otherwise the column 3 constant (i.e., CDOT) is used.

For an aromatic nitrogen with 3 bonds, column 3 is used for -N=+, column 4 for -N< in an aromatic ring which is not fused to another aromatic ring, and column 5 otherwise (phen-N<).

b) If the fragment is aliphatic, the choice of constant depends on the numbers and types of its attachments. The types of attachments which are determined and the abbreviations used are as follows.

PHEN	Primary (i.e., left-handed) attachment to an aromatic system
1/PHEN	right-handed attachment to an aromatic system
STYRYL	attachment to a carbon-carbon double bond which is directly conjugated to an aromatic system
VINYL	attachment to a carbon-carbon double bond which is not directly conjugated to an aromatic system
BENZYL	attachment to a carbon which is attached to an aromatic system

And the following notation is used for the constants from the fragment table.

	F	From	Col 2
	F(PH)	From	Col 3
	F(2PH)	From	Col 5
	F(1/PH)	From	Col 4
and	F(1R)	From	Col 6

If an aliphatic fragment has none of the above attachments, then the constant F is used.

If the fragment has only one of the above attachments, then the following values are used.

PHEN	F(PH)	
1/PHEN	F(1/PH)	
STYRYL	F	+.667[F(PH)-F]
VINYL	F	+.333[F(PH)-F]
BENZYL	F(1R)	

If there are 2 of the above attachments, the reference specifies the constant to be used in only the case of 2 PHENYL attachments. We have assumed the following conventions.

PHEN	PHEN	F(2PH)	
PHEN	1/PHEN	F(2PH)	
PHEN	STYRYL	F(PH)	+.667[F(2PH)-F(PH)]
PHEN	VINYL	F(PH)	+.333[F(2PH)-F(PH)]
PHEN	BENZYL	F(PH)	+.15 [F(2PH)-F(PH)]
1/PHEN	1/PHEN	F(2PH)	
1/PHEN	STYRYL	F(1/PH)	+.667[F(2PH)-F(1/PH)]
1/PHEN	VINYL	F(1/PH)	+.333[F(2PH)-F(1/PH)]
1/PHEN	BENZYL	F(1/PH)	+.15 [F(2PH)-F(1/PH)]
STYRYL	STYRYL	F	+.667[F(2PH)-F]
STYRYL	VINYL	F	+.5 [F(2PH)-F]
STYRYL	BENZYL	F	+.4 [F(2PH)-F]
VINYL	VINYL	F	+.333[F(2PH)-F]
VINYL	BENZYL	F	+.25 [F(2PH)-F]
BENZYL	BENZYL	F	+.15 [F(2PH)-F]

If there are more than 2 of these attachments, then F(2PH) is used.

Note that these calculations take the form of an interpolation. The computer implementation uses this form in all cases, using zero for the interpolation factor when there is no actual interpolation and taking appropriate constants from a set of arrays. As long as an interpolation scheme is adequate for fragment constant assignments, changes can be made by changing numbers in these arrays rather than by making programming changes.

11) Types and adds the assigned fragment values and makes proximity corrections. Each polar fragment which has been assigned a constant is considered for possible proximity corrections. Each leaving bond is checked for 1 and 2 carbon separation from another polar bond (the middle bond of a 2-carbon link cannot be polar to prevent the link from going through a fragment.) For each proximity occurrence the previously assigned fragment constant is multiplied by the appropriate proximity factor and this is added into the partition coefficient total. This is typed out separately. The appropriate proximity factor (aromatic ring, aliphatic ring, or chain) is based on the type of the leaving

bond in 1 carbon separations and on the type of the connecting carbon-carbon bond in 2 carbon separations. This is equivalent to finding proximity pairs in most cases since each link is found twice, once with each fragment involved, except that the factor type in 1 carbon separations depends on the leaving bond type and could be different for each "end".

To prevent redundant proximity corrections, the number of corrections is limited to twice the number of polar fragments in the molecule. Further, only one 1-carbon separation correction is allowed per connecting bond per fragment, only one 2-carbon separation correction is allowed per connecting bond per fragment, and a bond may be used as a middle bond in 2-carbon separation corrections only twice (i.e., for 1 proximity pair).

The corrections are made in ascending order of fragment constant values. (i.e., the most negative values first)

12) The partition coefficient total value is labeled "partial" if:

a) There is an empty value in the fragment table corresponding to any found fragment. (This will show in the constant column as "empty").
or

b) There are any atoms in the molecule which were not found as parts of fragments or as aliphatic carbons or hydrogens. (The atom indexes of these atoms are typed out).
or

c) There are both polar fragments and halogens in the molecule. (in this case a warning of possible H/S polar interractions is typed out).

Fragment Table

PUBLIC LOGPFRAGS is a table containing data from page 23 and appendix iii of the reference. It contains the fragments and constants used in the partition coefficient calculation. Column 0 contains the names of the fragments. Column 1 contains the fragments. These are prophet molecules with ? as the atom type for connecting atoms. Atom 1 in each fragment (i.e., the first atom drawn in) is assumed to be the primary connection. This is used in determining which column of fragment constants to use. If a fragment has symmetrical connections, one of which is the primary connection (e.g., -O-) then column 4 should have the same entry as column 3.

The table is divided into aromatic and aliphatic sections and these sections are further divided by elements to eliminate unnecessary searching. The fragments are ordered to insure that "larger" fragments will be found before the fragments they contain. (e.g., -OH precedes -O-) that is, any fragment which contains another should precede the one it contains.

It is assumed that all fragments with no R bonds precede the fragments with R bonds and that aromatic fragments are distinguished from aliphatic fragments by a name (in col 0) beginning with the letters ARs given in col 'SECTION'.

Columns 2 through 6 contain fragment constants. The column headings for these columns apply to aliphatic fragments. For aromatic fragments all constants are in column 3 with the exception of the entries for nitrogens and ring fusion carbons.

For arom N and arom NH the value in column 3 corresponds to -N=+, the value in column 4 to -N< which is in a aromatic ring which is not fused to another aromatic ring, and the value in column 5 to -N< which is in an aromatic ring which is fused to another aromatic ring.

For ring fusion carbons the value in column 3 corresponds to C-dot and that in column 4 to C-star.

Column 7 contains a MOLSCREEN of the fragment.

Column 8 indicates whether or not the leaving bonds of the fragments should be considered polar. This is used in determining group branches, ring clusters, proximity corrections, etc.

Currently, LOGPFRAGS lacks an entry for an aliphatic -N= fragment.